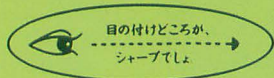
1994年8月1日発行（毎月1回1日発行）第13巻8号通巻148号 昭和58年11月2日第三種郵便物認可

PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

特集 Graphic Movement

レンズフレアのシミュレーション/PICTを使う/SX-PICSLICE/壁紙動画

完成間近！ X68000用68030アクセラレータボード

新製品紹介 SX-WINDOW ver.3.1/Xsimm VI/Mu-1 GS

1994

SOFT BANK オー/エックス
定価600円

フィールドが、膨らむ。

先が、ますます面白くなる。

●

未来への確かなビジョンをベースに
発展性のあるプラットホームとしてのウィンドウ環境を提供する
国産オリジナルウィンドウシステムSX-WINDOW。

●

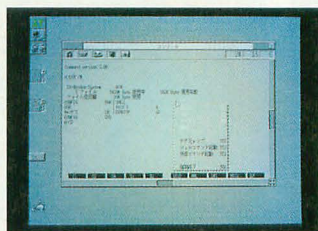
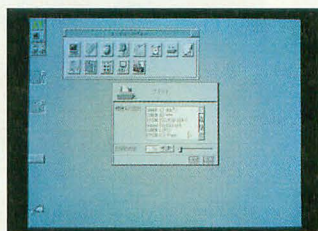
GUI環境や操作環境、高速化へのゆるぎない探求、
マルチメディアの統合的なハンドリング。

●

いま、より多彩なフィールドへ
そのインテリジェンスが展開を始める。

●

次のステージが見えてくる。

●インライン入力のサポート：ASK68K Ver.3.0を利用したインライン入力をSX-WINDOWで実行可能。またシャーペン.Xをワープロとして利用できるよう、さまざまな機能が付加されています。

●コンソールをサポート：Human68kやX-BASICのコマンドをSX-WINDOWアプリケーションと同時にタイムシェアリングで実行できます。
（グラフィックを利用したものなど、SX-WINDOWと処理が重複するものは実行できません。）

●多彩なプリンタに対応：さまざまなSX-WINDOWアプリケーションで利用できるページプリンタドライバを標準装備。ESC/Page、LIPS III、PostScriptに対応したプリンタが利用できます。

SX-WINDOW ver.3.1の
データ利用環境

今も、先も楽しめる。
いつも新展開の予感、SX-WINDOWのニューバージョン。

SX-WINDOW ver.3.1

「SX-WINDOW ver3.1システムキット」CZ-296SS（130㎜FD）/CZ-296SSC（90㎜FD） 標準価格22,800円（税別）

SX-WINDOW ver.3.1 発売記念

「シャーペン・カスタマイズコンテスト」のお知らせ

SX-WINDOW ver.3.1の発売を記念し「シャーペン・カスタマイズコンテスト」を実施します。あなたが一番使いやすい、世の中に知ってもらいたい、そんなシャーペン・カスタマイズの力作をどしどしお寄せください。

●応募要項●
あなたがカスタマイズした「シャーペン.ENV」と、簡単な説明をフロッピーディスクに入れ、EXE会員番号・住所・氏名・年齢を明記の上、下記住所まで送付ください。

●応募締切●
平成六年八月末日消印有効

●応募資格●
X68000/X68030EXEクラブ会員の方に限定させていただきます。

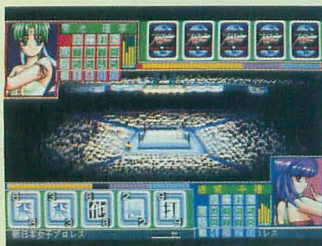
●特典●
◎最優秀作品には「ご希望の増設メモリボード・モジュール×1」「ご希望のSX-WINDOW対応ソフト×1」「インテリジェントコントローラCZ-8NJ2」のうちいずれか1点を進呈。
◎優秀作品は、今秋発行予定の「EXEディスク2」に掲載、ご紹介します。
（著作権は作者の方々に帰属します。）さらに、応募者全員にSX-WINDOWオリジナルSOSINA進呈。

●送付・問い合わせ先●
〒545 大阪市阿倍野区長池町22-22シャープ株式会社
電子機器事業本部システム機器営業部EXEクラブ事務局
TEL 06-621-1221（大代表）

シャープ株式会社

資料請求券
ペリフェラル
Oh!X
8係

特集 Graphic Movement

Mr.Do!/Mr.Do! vs UNICORNS

レッスルエンジェルス3

アクセラレータを作る

Mu-I GS

(で)のショートプロぱーてぃ

Oh!X

CONT

●特集

●カラー紹介

●THE SOFTOUCH

●シリーズ全機種共通システム

〈スタッフ〉
●編集長/前田 徹 ●副編集長/植木章夫 ●編集/山田純二 豊浦史子 高橋恒行 ●協力/有田隆也 中森 章 林 一樹 吉田幸一 華門真人 朝倉祐二 大和 哲 村田敏幸 丹 明彦 三沢和彦 長沢淳博 司馬 護 清瀬栄介 石上達也 柴田 淳 瀧 康史 横内威至 進藤慶到 ●カメラ/杉山和美 ●イラスト/山田晴久 江口響子 高橋哲史 川原由唯 ●アートディレクター/島村勝頼 ●レイアウト/元木昌子 ADGREEN ●校正/グループごじら

表紙絵：塚田 哲也

1994 AUG. 8

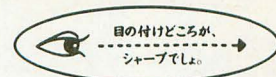
UNIXはAT＆T BELL LABORATORIESのOS名です。
Machはカーネギーメロン大学のOS名です。
CP/M, P-CPM, CP/Mupis, CP/M-86 CP/M-68K, CP/M-8000, DR-DOSはデジタルリサーチ
OS/2はIBM
MS-DOS, MS-OS/2, XENIX, MACRO80, MS C, WindowsはMICROSOFT
MSX-DOSはアスキー
OS-9, OS-9/68000, OS-9000, MW CはMICROWARE
UCSD p-systemはカリフォルニア大学理事会
TURBO PASCAL, TURBO C, SIDEKICKはBORLAND INTER NATIONAL
LSI CはLSI JAPAN
HuBASICはハドソンソフト
の商標です。その他，プログラム名，CPU名は一般に各メーカーの登録商標です。本文中では"TM"，"R"マークは明記していません。

E N T S

●読みもの

●連載/紹介/講座/プログラム

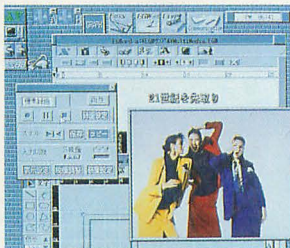
■広告目次

ビデオグラフィックスの
世界へ。
資料請求券
X68030
■お問い合わせは… シャープ株式会社
電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

1,677万色対応、ビデオ映像を高画質・高速取り込み

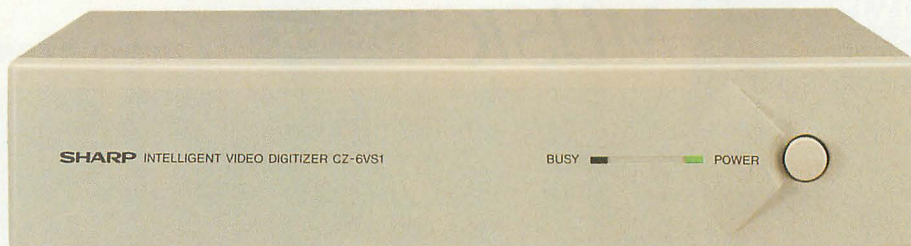
テレビやビデオ、ビデオディスクなどの映像をX68シリーズやMacシリーズ※1の動画・静止画データとして高速取り込みが可能、いわば"ビデオスキャナ"とでも呼びたいビデオ入力ユニットです。1,677万色対応、最大640×480ドットの高解像度※2。動画・静止画の手軽なハンドリングが、新たなグラフィックシーンを創造します。

※1 MacintoshはIIシリーズ以降の機種に対応、ディスプレイ解像度が640×480ドットの場合、取り込み可能な範囲は、160×120ドット、320×240ドットのサイズになります。
※2 X68030/X68000シリーズでは、1,677万色はデータ作成のみに対応。表示は最大65,536色、解像度は512×512ドット。また、Macintoshは機種により表示色数が異なります。

アプリケーションツール「ライブスキャン」を標準装備

動画や静止画を簡単に保存できるアプリケーションソフト「ライブスキャン」※を標準装備。取り込んでいる映像を表示したり、残したいシーンを簡単に静止画保存したり、手軽な動画・静止画ハンドリングでパソコンの可能性をさらに広げます。X68030/X68000シリーズ用SX-WINDOW対応版とMacintoshシリーズ用QuickTime対応版の2種類を同梱しています。

※SX-WINDOW版はバージョン3.0以降（メモリー4MB以上）、QuickTime版はMacintosh漢字Talk7リリース7.1以上のシステムとQuickTime1.5以上（メモリー8MB以上）が必要です。

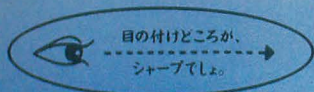
1,677万色対応の高速映像取り込み、動画・静止画の手軽なハンドリングが、新たなマルチメディアシーンを創造する。

SHARP INTELLIGENT VIDEO DIGITIZER CZ-6VS1　BUSY　POWER

■SCSIインターフェイス採用：パソコンの専用I/Oスロットを使わずに接続可能になり、汎用化を実現しました。またSCSI-2（FAST）インターフェイスの採用により、データ転送速度の高速化を図っています。X68030/X68000シリーズでは、SCSI-2（FAST）対応のハードディスクを接続することにより、パソコン本体を経由しないで、ハードディスクに直接、動画データをテンポラリデータとして記録することが可能です。パソコン本体のハードディスクへは、記録終了後に、テンポラリデータを変換し動画データとして保存できます。

※CZ-600C/601C/611C/602C/612C/652C/662C/603C/613C/653C/663Cに接続する場合は別売のSCSIインターフェイスボードCZ-6BS1ならびにSCSI変換ケーブルCZ-6CS1が必要です。※CZ-604C/623C/634C/644Cに接続する場合は、別売のSCSI変換ケーブルCZ-6CS1が必要です。
※Macintosh Power Bookシリーズに接続する場合は別売のSCSIケーブルなどが必要です。詳しくはMacintosh Power Bookシリーズの取扱説明書をご覧ください。

■高機能MPUを搭載：クロック周波数25MHzの32ビットMPU/MC68EC020を搭載、高速処理やパソコン本体の負担の軽減を実現します。

●MacはMacintoshの略称です。●Macintosh、Macintosh IIは、米国アップルコンピュータ社の登録商標です。●Power Bookは米国アップルコンピュータ社の商標です。●漢字Talk7はアップルコンピュータジャパン社の商標です。●QuickTimeは、米国アップルコンピュータ社の商標です。●価格には、消費税及び配送・設置・付帯工事費、使用済み商品の引き取り費等は含まれておりません。

標準価格178,000円（税別）

For X68030/X68000series APPLICATION SOFTWARE

◎パーソナルDTPをX68で

CZ-291BWD 標準価格35,000円(税別)

NEW

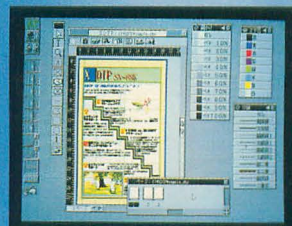
縦書きをはじめとした多彩なレイアウト機能で
パーソナルなデスクトップパブリッシングを実現するソフトです。
やさしい操作、豊富な編集機能、グラフィックウィンドウ対応、SX-WINDOWをすでに
ご利用になっている方なら、基本操作を新たに覚えることなく手軽にレイアウトが作成できます。

■豊富なテキスト編集機能：フォント種類、サイズ、文字種の変更はもちろん、上線、下線、網掛け、文字間隔の指定が文字ごとに設定可能。禁則、行間隔、タブ、インデント、マージンもパラグラフ(リターンコードまでの文字列)ごとに設定できます。また各テキストフレームごとに、フレーム形状、リンク状態(テキストの流し込み)、縦書き/横書き、回り込みの設定が可能。検索/置換も単純な文字列だけでなく、スタイル別に行うことができます。

■グラフィックウィンドウに対応：GRW.Xにも対応していますので、いろいろな形状でレイアウトしたグラフィックフレームのデータを65,536色の画像で確認しながらレイアウトできます。

■さまざまな画像フォーマットに対応：ビデオマネージャが対応している静止画フォーマットの他に、「PrintShop PRO-68K」、「CANVAS PRO-68K」、「GScriptファイル」の読み込みに対応しています。

●グラフィックフレーム、テキストフレームとは別に直線、矩形、楕円、多角形が作成できる独立した罫線機能 ●第1水準を収めた明朝体、ゴシック体のベジェーフォントファイルを標準装備 ●ページの移動や作成/削除がスピーディに行える独立したページウィンドウをサポート ●ページプリンタドライバ(ESC/Page、LIPSⅢ)を付属、高解像度の美しい印字が可能。またSX-WINDOWが対応しているプリンタも使用可能。

※5MB以上の空きのあるハードディスクが必要です。 4MB、Ver.3.0

◎グラフィック感覚の楽譜入力をサポート

MUSIC SX-68K CZ-274MWD 標準価格38,000円(税別)

NEW

MIDI、FM、ADPCMに対応した楽譜ワープロ&作曲演奏ソフトです。
自由なレイアウトでグラフィックを描くように楽譜入力、
全パートの同時入力や編集、自動伴奏機能、応用範囲を広げるデータ互換性。
多彩なプリンタ対応で美しい印刷も可能です。

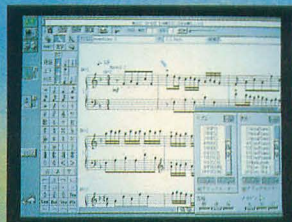
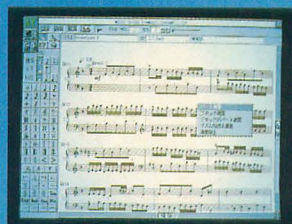
■MIDI、FM、ADPCM対応：MIDI、FM、ADPCMを同時に発音できます。全ての音源を利用した場合、最大発音数は25まで設定可能です。

■全パートの同時入力：ピアノ譜、メロディ譜などの組み合わせで最大16パートまで編集可能。特定パートごとではなく全パートを画面に表示して編集できますので、直接画面上で曲の構成を考えながら作編曲できます。

■コード&リズムによる自動伴奏機能：メロディ上にコードネームとリズムパターンを入力するだけで、自動的に伴奏をつけることができます。

■優れたデータ互換性：「MUSIC PRO-68K」、「MUSIC PRO-68K[MIDI]」のデータファイルが利用できる他、OPM、MML、ZMSファイル形式でデータ出力が可能です。

■多彩なプリンタ対応：ページプリンタドライバ(ESC/Page、LIPSⅢ)を付属、高解像度の美しい印刷が可能です。
またSX-WINDOWが対応しているプリンタも利用できます。

4MB、Ver.3.0

その先のシーンへ。

●さらに実用的なウィンドウシステムへの進化

SX-WINDOW ver3.1 システムキット

CZ-296SS(130㎜FD)/CZ-296SSC(90㎜FD) 標準価格22,800円(税別) NEW

ASK68K Ver3.0を利用したインライン入力のサポート、Human68k/BASICコマンドをSX-WINDOWアプリケーションと同時にタイムシェアリングで実行できるコンソールのサポートをはじめ、シャーペン.Xをワープロとして利用できるよう機能アップ。また、さまざまなSX-WINDOWアプリケーションで利用できるページプリンタドライバを標準装備。ドローデータ(FSX)/フォントデータ(IFM)処理の高速化も実現しています。

※コンソールでは、SX-WINDOWと処理が重複するものは実行できません。 4MB

※既にSX-WINDOWをお持ちの方には有償バージョンアップサービスを行います。

●定評のGUI対応ウィンドウワープロ

EGWord SX-68K

CZ-271BWD 標準価格59,800円(税別) NEW

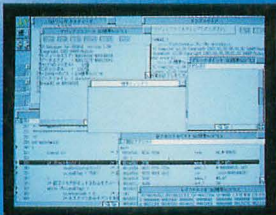
ウィンドウワープロとして評価の高いEGWordのSX-WINDOW対応版。キャラクタベースのワープロを超えたグラフィカルユーザーインターフェイス(GUI)による手軽なDTPソフトとしても優れた表現力を発揮します。定評ある日本語入力方式(EGConvert)によるインライン入力、さまざまなグラフィックデータ(GScript)やテキストデータの貼り込み、また文書互換を実現するEDF(Extended Document Format)形式をサポートしています。 4MB、ver.2.0

●SX-WINDOW開発支援ツール

SX-WINDOW 開発キット Workroom SX-68K

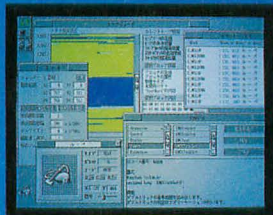
CZ-288LWD 標準価格39,800円(税別) NEW

SX-WINDOW用のソフト開発に必要なツールやサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業をSX-WINDOW上で効率よく実行できます。初めてSX-WINDOW用のプログラムに挑戦する人にも、簡単に基本機能の理解が深まる33種(基礎編23種、応用編4種、実用編6種)のサンプルプログラム付き。

※ご使用に当ってはC compiler PRO-68K ver.2.1が必要です。 4MB、ver.2.0

●SX-WINDOW開発キットのサポートツール

開発キット用ツール集

CZ-289TWD 標準価格12,800円(税別) NEW

SX-WINDOW開発キットをさらに使いやすくするためのツールです。SXコールの簡易リファレンスを簡単に検索するインサイドSX、イベントの発生を常時監視・確認するイベントハンドラ、リアルタイムにメモリブロックの利用状況を表示するヒープビューアなど11種のツールが用意されています。 2MB、ver.2.0

●SX-WINDOW対応ドローイングツール

Easydraw SX-68K

CZ-264GWD 標準価格19,800円(税別)

イラスト、フローチャート、地図、見取り図など各種グラフィックが製図感覚で作成できます。作成したデータは他のSX-WINDOW対応アプリケーションでも利用でき、企画書などの作成をサポート。ページプリンタドライバも標準装備。 4MB、ver.3.0

●ウィンドウ対応グラフィックツール

Easypaint SX-68K

CZ-263GWD 標準価格12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応ペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でのデータ交換もできます。 2MB、ver.1.1

●SX-WINDOWを楽しく使うためのアクセサリ集

SX-WINDOW デスクアクセサリ集

CZ-290TWD 標準価格14,800円(税別)

SX-WINDOWをさらに便利に楽しく使うためのデスクアクセサリ集です。スクリーンセーバ、スクラップブック、スケジューラ、アドレス帳、電子手帳通信ツール、パズルなど、12種の豊富なアクセサリが収められています。 2MB、ver.3.0

●マルチタスク機能をはじめ通信環境がさらに充実

Communication SX-68K

CZ-272CWD 標準価格19,800円(税別)

通信環境をさらに高めたウィンドウ対応の通信ソフトです。マルチタスク機能により他のアプリケーションを実行中でも簡単に通信が可能。自動ログイン機能やプログラム機能、など豊富な機能をサポートしています。 2MB、ver.1.1

●FM音源サウンドエディタ

SOUND SX-68K

CZ-275MWD 標準価格15,800円(税別)

他のミュージックソフトで演奏中の音色を、簡単に作成、変更できるマルチタスク機能、またエディット、イメージ、ウェーブの3つの編集/確認モードを装備。作成中の音色も50曲の自動演奏でリアルタイムに確認、編集できます。 2MB、ver.1.1

●SX-WINDOW対応になってさらにパワーアップ

倉庫番リベンジ SX-68K ユーザー逆襲編

CZ-293A(130mmFD)/CZ-293AWC(90mmFD) 各標準価格6,800円(税別)

倉庫番10年にわたるユーザーの投稿など、新作306面が目白押し。まさに倉庫番の最強版がSX-WINDOW上で楽しめます。AI機能やエディット機能、キャラクタ変更機能も装備。半年で解けたらあなたは天才?です。 2MB、ver.1.1

PRO-68K シリーズ

●X68030/X68000対応

CZ-295LSD 標準価格44,800円(税別)

C compiler PRO-68KのX68030/X68000対応版。MPU68030、MC68882の命令セットに対応したアセンブラ、デバッガ、ソースコードデバッガを付属。またHuman68k ver.3.0、ASK68K ver.3.0にも対応。新たにGPIBライブラリ、MC68882対応フローートライブラリを付属しています。

※メインメモリ2MB以上が必要です。
※C compiler PRO-68K/ver.2.0/ver.2.1をお持ちの方には有償グレードアップサービスを行います。

※(2MB,ver.1.1)の表示は、メインメモリ2MB以上、SX-WINDOW ver.1.1以上が必要であることを示します。

※発売予定のソフトの画面は実物とは異なる場合があります。

●EGWord、EGConvertは株式会社エルゴソフトの登録商標です。●ESC/Pageはセイコーエプソン株式会社の登録商標です。

●お問い合わせは…シャープ㈱電子機器事業本部(液映)システム機器推進プロジェクトチーム 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)へ シャープ株式会社

資料請求券

SHARP

高速、高解像度。

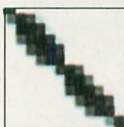
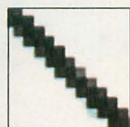
透過原稿・ADF対応型カラーイメージスキャナ、誕生。

●拡大読み取り時、細かい部分でも忠実に再現。
2400dpi※1やデジタルズーム機能が高品位を守ります。

●35ミリフィルムも透過原稿読み取りユニットを使用して読み取り可能。

高解像・高品位。美しさが際立ちます。

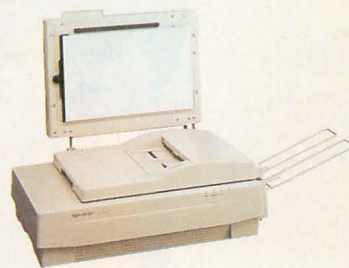
基本解像度600dpi、疑似解像度2400dpi※1の高解像度読み取りで微細な点や線を鮮明に再現します。縮少・拡大は30〜2400dpiの範囲で設定可能です。また、約1677万色で原画に忠実なリアルな色合いを再現します。

●シャープ独自の技術「デジタルズーム」搭載により繊細な線やズーム画像も忠実に再現。また「ワンウェイスキャン方式」を採用し、凹凸のある原稿も鮮明に読み取りできます。

通常の拡大時
(当社従来機)

デジタルズーム
(JX-330)

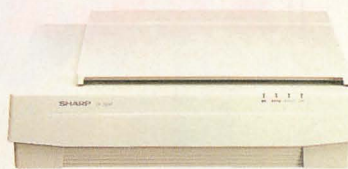
高速処理を実現。スピーディに作業できます。

A4、300dpiならカラー約13秒※2、モノクロ約1秒※2でこのクラス最高の※3高速読み取りが可能です。大きな画像データを高速転送できるSCSI-Ⅱにも対応。また、最大A4/リーガルサイズ(216.4×355.6mm)までの原稿を読み取りできます。

透過原稿読み取りユニットとADFを同時装着できます。

透過原稿読み取りユニットは、35mm(ネガまたはポジ)フィルムからレントゲン写真まで各種透過原稿※4に対応。基本解像度600dpi/1200dpiの2種類をご用意しました。また最大50枚までの原稿を自動送りできるADFも同時装着できます。

X68000対応カラーイメージスキャナ

JX-330X

透過原稿読み取りユニット(オプション)
JX-3F6 標準価格 98,000円(税別)
JX-3F12 標準価格138,000円(税別)

カラーイメージスキャナ
JX-330X 標準価格178,000円(税別)

ADF[原稿自動送り装置](オプション)
JX-AF3 標準価格 58,000円(税別)

使いやすい高機能画像入力ソフトを標準装備<JX-330X>
●Scanner Tool/S(画像入力ソフト)、対応フォーマット形式：ZIM、PIX、GL3、PIC、GLX、GLM

※1 2400dpiは当社独自手法による疑似解像度です。※2 読み取り開始から読み取り終了までの動作時間。ただし初期動作およびデータ転送時間を除く。※3 クラスとは、A4フラットベットクラスのこと。'94年4月現在。※4 読み取り可能なサイズは機種によって異なります。
■消費税及び配送・設置・付帯工事費・使用済み商品の引き取り費等は、標準価格には含まれておりません。

■資料のご請求・お問い合わせは シャープ株式会社 プリントシステム事業本部プリントシステム営業部
〒545 大阪市阿倍野区長池町22番22号 TEL.(06)621-1221(大代表) FAX.(06)629-1207
〒162 東京都新宿区市谷八幡町8番地 TEL.(03)3267-4410(ダイヤルイン) FAX.(03)3260-2159

シャープ株式会社

ゼッタイわかる!

Hello!

GAMEBLAST
10月8日創刊
©1993 Embodiment Films, AS Entertainment Planning & PIA

新刊

「ファー・ローズ・トゥ・ロード」リプレイ

RPGセッションガイド

遊演体 監修　司史生/ゆうせぶん 著　定価1,600円

国産テーブルトークRPGシリーズの最新作、「Far Roads to Lord」初の公式ガイドブック。リプレイを中心に、ルールのリファレンスや、背景世界ユルセルームの解説を盛り込み、「F·ローズ」のマスターおよびプレイヤーに、その魅力とプレイ方法を紹介しています。

近刊　7月下旬発売予定!

逆引きモンスターガイド～東洋編

ヘッドルーム 編著

東洋世界のモンスターたちにスポットをあてた、西洋編に続く第2弾。阿修羅、酒呑童子、かまいたち、河童、子泣き爺など、日本を含む東洋世界に起源を持つ神・妖怪・化物を多数取り上げ、女神転生、天外魔境、桃太郎伝説、ONIなどの人気RPGシリーズでの彼らの姿を紹介しています。　予価1,800円

RPG幻想事典シリーズ

好評発売中!

逆引きモンスターガイド～西洋編
ヘッドルーム編著　定価1,800円

戦士たちの時代
司史生/坂東いるか 共著　定価1,800円

チャンバラ英雄伝
柳川房彦/高井夏生/横山雄一 共著　定価1,800円

RPG幻想事典·日本編
飯島健男 著　定価1,860円

RPG幻想事典
早川浩 著　定価1,550円

●定価は税込みです　●お近くの書店でお求めください

ソフトバンク株式会社／出版事業部
販売局 TEL.03-5642-8101
SOFT BANK

Oh!X

Graphic Gallery

DōGA CGアニメーション講座

今月は「GENIE」の使い方の2回目です。先月号で作成したメカを使って，その動きをデザインしてみましょう。

●非常に簡単な動きの例として，このようなモーションをデザインします。遠くから，まっすぐこちらに向かって飛んでくるだけです。データは10フレームで構成されています

●光源の向きを変更すると雰囲気が変わる

基本のライティング。右斜め上から光が当たっていて，適当に影もできる

光線が視線と完全に一致した場合。影ができないので，メリハリがない

逆光。どんな形か全然わからないが，演出としては面白い効果も出せる

●同じパーツをベースにしても，作り方によって戦闘機にも戦艦にもなる

ベースにするパーツ(DS05)

戦闘機にした例。戦闘機は，コックピット，ハネを大きくし，1～2門のキャノンをつけるとそれらしくなる

戦艦にした例。小さな艦橋をつけるとなんでも戦艦になる。巨大感を出すためには，一部だけディテールを細かくすればよい

「SWORD」森山さんの恐竜

「SWORD」「SWORD 2」でグランプリを受賞した森山氏は，第6回に出品していないと思ったらこんなCGAを作って遊んでいました。ちょっと見には，CGAシステムで作られたとは信じられませんね。これが継ぎ目もなく，なめらかに動く様子は，まさに「ジュラシックパーク」。でも，ちょっと凝りすぎなので，この調子で何か作品を作るのは無理でしょうねえ

KMCのゲーム「ドリーミング」

KMC(京大マイコンクラブ)が制作したパズルゲーム。TAKERUで販売します。詳しくは114ページをご覧ください

THE USER'S WORKS

●HELL HOUND

実に久しぶりのUSER'S WORKS。今回紹介するのはX68000ユーザーには馴染み深いグラディウスタイプのシューティングゲームだ。X68030ならAD PCM多重で起動するぞ。

左はタイトルとウェポンセレクト画面。レーザー系兵器とミサイル系兵器をそれぞれ２つずつ選んで持っていく。武器はカプセルで切り換える。

●

画面を見てわかるとおり，非常にトラディショナルな構成の横スクロールシューティングゲームである。投稿原稿にも「グラディウスタイプの横スクロールシューティングゲーム」とあるが，ところどころ，一連のグラディウスシリーズを知っている人なら思わずニヤリとするようなところがある。多大な影響は随所に見られる。

そう考えれば面構成や操作感覚も非常にわかりやすいのだが，オリジナリティに欠けるという点ではもの足りない感じもしなくはない。うーむ。

無論，グラディウスそのままではなく，ボタンによりオプション（このゲームではトレーサと呼ぶ。もちろん４つまで）が固定できたり，ダメージ制を採用したり，ミス時のパワーダウンが少なくなっているなどの点で本家とはまた違った攻略性も備えている。

ディスク1枚に収めているのはよいが，面構成が５面というのがやや少なく感じられる。思わず「高速スクロール面！」と身構える場面が実は最終面だったので余計そう思えるのかもしれないが。

プログラムは全体的によく動いている。グラフィックは非常によい。音楽もなかなか頑張っているのだが（内蔵音源とSC-55に対応），フュージョン系の曲は雰囲気にあっているかという点では意見が分かれるところだろう。それっぽいボコーダボイスでしゃべるのもいい感じだ。

最大の欠点は当たり判定が大きいこと。通常，この手のゲームは自機の当たり判定が非常に小さいものなのだが，やっていて障害物の当たり判定が外枠の矩形で取られていることがよくわかる（弾との判定は小さい）。グラディウスシリーズだったら，こんな大きな障害判定なんてネメシスくらいのもんだぞ。

ボスキャラとの当たり判定などでは，露骨な安全地帯（？）もあるが，まあ，それも善し悪しかもしれない。全体の難易度は中級といったところだ。

セミオートのパワーアップシステムがいまいちわかりにくいのも難点か。個人的にはスピードアップは別個にしてほしかったところだ。

このゲームを入手希望の方は，無記名定額小為替2000円分（送料込み）と返信用の宛名シールを郵送して下記住所まで連絡を取ってほしい。なお，通信販売の受付は1995年２月いっぱいまでとなっているので注意すること。

<連絡先>

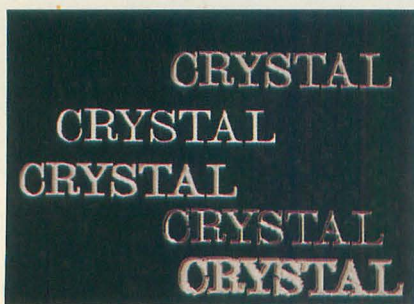
〒165　東京都中野区沼袋3-6-4-101

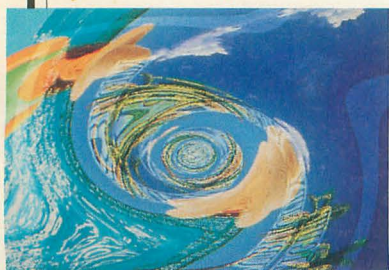
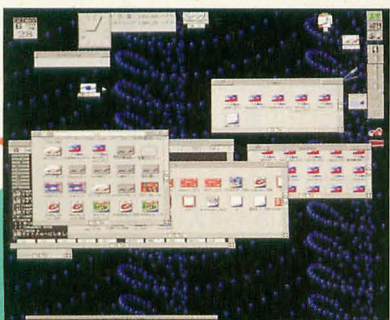
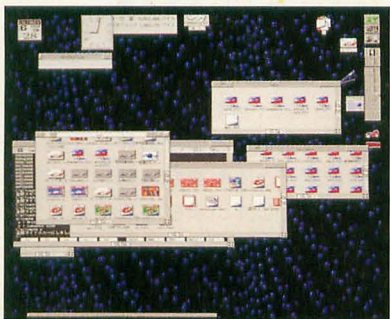
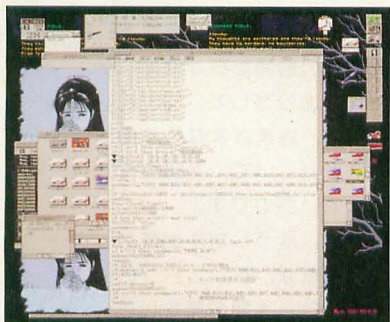
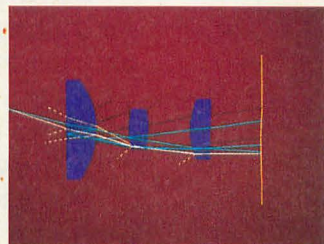
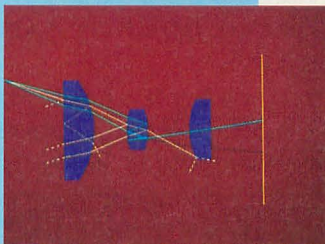
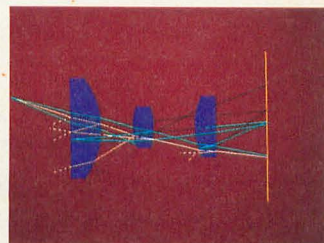
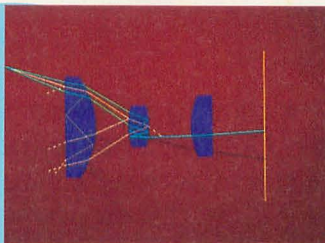
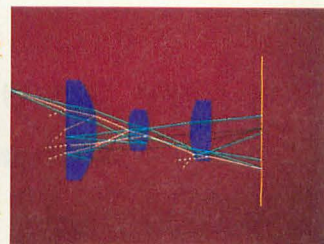
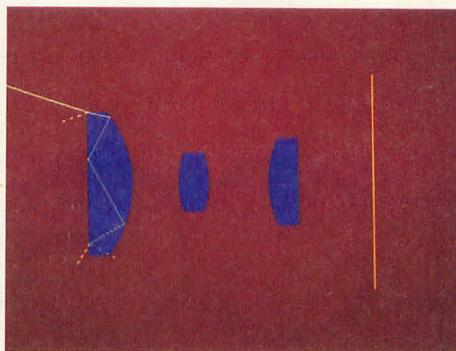
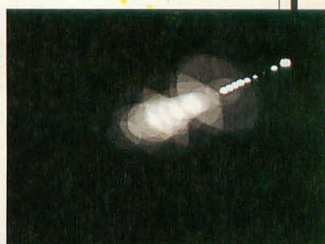
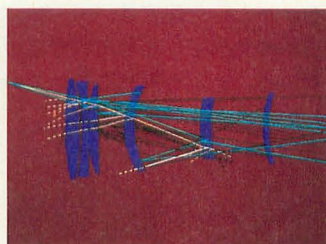
小松方B/T係

これが各面のようすだ。ざっと紹介しよう。１面は遺跡（？）面，２面は内臓面，３面は逆火山，４面は宇宙面，そしてラストステージは要塞面となる。各ステージの終わりには，もちろんボスキャラ（または相当品）が待ちかまえている。それにしてもサーチレーザーは使えない……。

[特集]

Graphic Movement

ひととおりのことをやり尽くされた感のあるグラフィック処理
しかし，まだまだ探せばこれまでできなかった処理，
手をつけられていない分野はいっぱいあります
そしてSX-WINDOW上でのグラフィック処理の展開はこれから始まるのです
これまでの方向とはちょっと趣を変えた処理を探ってみましょう

ガラス化処理。左は，まずはロゴ文字を加工してみたもの。上から2番目が元の文字。右は，画像の上に合成したもの（下段は透明度つき）。下は普通の画像にガラス化処理を加えてみたところだ。

4枚のPIC画像を同時に画面にロードしているところ。速度の差は一度に展開するライン数をいろいろ変えているから。

色のチャネルごとに分離して処理を加える。上のように変形した画像の色相成分を元画像に合成したところ。右はMATIERのオートペイントで作成された画像に元画像の輝度情報をだけを抽出して合成してみたところだ。

いわゆるブロンドのおねーさんとか，背景として活用（?）されるグラフィック画面。SX-PICSLICEを使えばマルチタスクのまま背景を切り替えることも可能だ。さらに背景を動かす壁紙動画と壁動玉々も強力だ。

設定されたレンズ面上に光線が反射/屈折していく様を追ってみる。レンズ同士の反射によってフィルム上に結ぶ予定外の光源映像，それがレンズフレアだ。このレンズ設定によるフレア例が下の写真にあたる。

上のように設定されたレンズ群によって生まれるレンズフレアのシミュレーション映像。光が円盤上に連なっている。いわば「光源の影」のようなものだ。右の写真は生成された画像をMATIERで合成したもの。

[第39回]

一角獣幻視

江口　響子

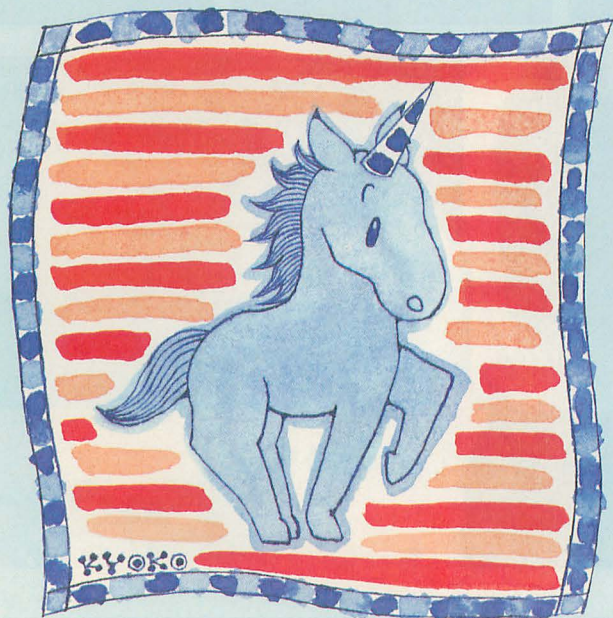
響子inCGわ～るど

CGでチェスの馬に角をつけて一角獣を作ったところ，それを見た友人のO氏が，題名は忘れたけれどルイ・マル監督のフランス映画の冒頭にこんなシーンがあったよと，話してくれました。

一角獣は，深い緑の森にひっそりと棲んでいる。あまりにも臆病で恥ずかしがり屋なので，人前にはなかなか姿を現さない。

でも，ひとつだけ一角獣を捕える方法がある……それは，ひとりの汚れなき美しい乙女を森に行かせて，木々の間に座らせるのだ。しばらくすると，一角獣がやってきて，彼女のひざにちょこんと頭をのせて寝てしまうだろう。そこを，つかまえればよい。

なぜ清純な乙女のところに一角獣がやってくるのかがちょっとした謎です。もしかしたら，疑うことを知らぬ純粋無垢な心を持つ者にのみファンタジーは訪れる，という寓話になっているのかもしれません。

*　　　　*

いまでは，一角獣は空想のなかだけに存在する生き物ですが，こうした空想の動物たちが現実にいると信じられていた時代もありました。19世紀以降に写真が普及するまでは，さまざまな生き物たちが図鑑にイラストで描かれていて，実際にその形で生存すると考えられていたのです。

荒俣宏さんの『怪物誌』には，そうした生き物たちが集められています。一角獣は，ヨンストンの「禽獣虫魚図譜」に出ていますし，人間の顔をしたライオンのスフィンクスもエジプトにいる生き物ということになっていました。また，カバやキリンは，実物とは似ても似つかぬ形で紹介されています。探検に行って遭遇した生き物を，目撃者の証言をもとに描いているのですが，記憶のあいまいな部分を空想や願望で補った結果，とても奇妙な動物や植物が存在することになってしまいました。

いまでは，動物園や植物園に行けば，図鑑にのっている生き物たちをこの目で見ることができます。また，実物を見ることができなくても，写真やテレビに写っていれば，ああ確かに存在するのだと信じ，それ以外は，伝説やファンタジーとして片づけがちです。でも，もしかしたら存在するのではと考えるのはやはり楽しい。ヒマラヤの雪男。東北地方の座敷わらし。そういえば昔，UFOから降り立った銀色の小さな宇宙人の有名な写真を，子供向けの本で見たことがありますが，あの宇宙人はその後どうなったのでしょうか。

*　　　　*

さて，清純な乙女のひざに頭をのせるというのですから，この一角獣はたぶんオスでしょう。では，一角獣のメスをつかまえるには，どうしたらいいのでしょうか。

美少年かそれともムキムキの筋肉男か……美少年にしても，あんまりナヨナヨしたのは気持ちが悪いし，かといって筋肉がムキムキすぎてもやっぱり気持ち悪い～……となると，すべてにバラン

スのとれた王子様タイプが妥当な線なのかなあ……。

何をそんなに悩んでいるの，簡単じゃないかとO氏。いわく，そもそも角は動物のオスに生えているもので，メスにはたぶん生えていない。それなら外見はただの馬だし，捕らえても希少価値はない，つまり捕らえるという発想自体が意味をなさないんじゃないか，と。

なるほど……ね。

＊『怪物誌　Monster Makers』荒俣宏著　ファンタスティックDOZEN12巻，リブロポート刊，2,060円(税込)

今回のCGデータ

1280×1024ピクセル
1670万色フルカラーを４×５ポジで出力
使用ソフトはサイクロン
総物体数136(物体数106，論理演算30)
平行光線１
マッピングデータは，著作権フリーの素材データ集から
「CGわ〜るど」のCGは，ピクセルの縦横比（アスペクト比）１：１でレンダリングしたものを，ポジ出力しています

SOFTWARE INFORMATION

みんなの希望の声が届いたようで，今月は新作情報が３本やってきました。「Ｘじゃなくて残念」とか「２がいいな」なんてことはいわないように。発売はまだ先だけどいい子にして待っていてね。

餓狼伝説SPECIAL

いよいよ発売目前となった「餓狼伝説SPECIAL」。ご存じのように，NEO・GEO版は去年の夏休み頃に発売されている。前作「餓狼伝説２」とは，見かけはあまり変わらないように見えるけれど，内容は変わっている。

まず，動きが軽快に，そして速くなっている。これについては，前作では本物より多少遅かったX68000版は今回どこまでがんばってくれるか楽しみである。

そして，キャラクターが増え，実に15人，いや16人である。最後のひとりを除く15人は最初から使うことができる。15人ものキャラを極める時間を考えれば，ゲーセンではもうすっかり姿を消してしまったけど，まだまだ十分遊べるゲームといえるだろう。

大きい変化はこの２点だが，細かい部分の変更点もいろいろあるので，お楽しみに。

あとは，X68000版をどこまでうまく移植してくれるかに期待。「餓狼伝説２」の移植度は，私には少し不満が残っている。ということで，参考にするNEO・GEO版を編集部に持ち込んで気合を入れて待っている私である。

来月号の発売までには店頭に並んでいるはずで，Oh!Xでも評価版によるレビューをお届けできるだろう。（瀧）

X68000用　5″2HD版　価格未定
魔法株式会社　☎078(261)2790

ビッグウエーブの前の静けさかな

		(前回順位)
1.	餓狼伝説SPECIAL	2
2.	XDTP	3
3.	スタークルーザーII	4
4.	サムライスピリッツ	5
5.	SX-WINDOW. ver.3.1	1
6.	Mr.Do!/Mr.Do! vs UNICORNS	5
7.	Mu-1 GS	−
	魔法大作戦	7
9.	X CASE	−
10.	クイーン・オブ・デュエリスト外伝＋α	−

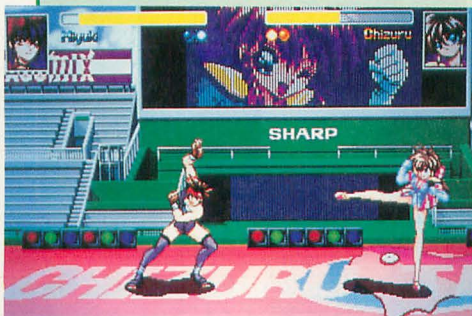
７月号の読者アンケートはがきの「期待の新作ソフト」を集計したものです。

前回１位の「SX-WINDOW ver.3.1」が５位になっていますが，これはすでに発売ずみで評判もなかなかよいようです。それについては28ページを見てくださいね。

で，発売を目前に控えた「餓狼伝説SPECIAL」への期待がいよいよ大きくなってきました。ちなみに２位に２倍以上の差をつけるという強さ。発売開始は７月28日の予定です。

６位「Mr.Do!/Mr.Do! vs UNICORNS」と７位の「Mu-1 GS」は最初の予定よりも発売が延びてしまいましたが，どちらもこの号が出る頃には発売されているはずです。評価版での製品レビューはそれぞれ24ページと68ページに掲載されています。

さて，今月の新作情報のうち初登場のものは３本ありますが，来月はこれらへの反響も期待されるところです。特に「スーパーストリートファイターII」と「プリンセスメーカー」は知名度も高く，編集部へのはがきのなかでも移植希望の声が多く寄せられていたものです。どちらも発売日など詳しいことは未定ですが，楽しみですね。

クイーン・オブ・デュエリスト外伝＋α

先月号で初めてお知らせしたこのゲームだが，X68000版の開発中の画面が入手できた。背景はすべて描き換えるとのことだったが，その証拠(？)に，この写真には「SHARP」の文字が堂々と……。

いま現在ではソフトの試用版は届いていないので，詳細は残念ながらまだ不明。PC-98，FM TOWNS版では「外伝」「外伝α」と重ねて発売されてきたが，X68000版ではひとつにまとめたものを強化との話である。

登場キャラは8人の美少女たち。発売は8月の予定なので，「やっぱりゲームは女の子でなくちゃ」という人はもう少し待っていてね。

X68000用　3.5/5″2HD版　5,800円(税込)
TAKERU　☎052(824)2493

スーパーストリートファイターII

カプコンの新作決定！　新キャラ4人の登場で，ますますバトルはアツくなる！

アーケード版との違いは，対コンピュータ戦とプレイヤー2人の対戦に加え，8人枠トーナメントモードが1台で実現されている。この勝ち抜き戦にはプレイヤーは最大8人参加できる。

推奨16MHz以上で，必要メモリは4Mバイト。

発売は9月の予定である。

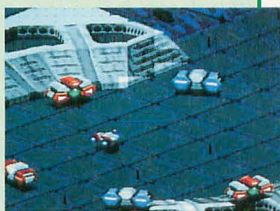
X68000用　5″2HD版　価格未定
カプコン

VIEW POINT

このゲームはクォータービューの斜めスクロールシューティングゲームである。1992年にNEO・GEO，昨年FM TOWNS版が発売されているのでご存じの方も多いだろう。

高い難易度で，マニア受けするゲームだがそのへんはX68000版ではどうなるだろか。

敵キャラは魚やら芋虫やらと多彩で，これらがみな，くねくね動くのだ。全体的になんか可愛いキャラクターたちである。

X68000用　5″2HD版　価格未定
ネクサス インターラクト　☎03(5474)3581

発売中のソフト

★Mu-1 GS　サンワード
X68000用　5″2HD版　28,000円(税別)
★Mr.Do！/Mr.Do！ vs UNICORNS　電波新聞社　7/2
X68000用　5″2HD版　5,900円(税別)
★宝魔ハンターライム12　TAKERU　7/10
X68000用　3.5/5″2HD版　1,500円(税込)
★餓狼伝説SPECIAL　魔法株式会社　7/28
X68000用　5″2HD版　9,800円(税別)
★レッスルエンジェルス3　TAKERU　7/31
X68000用　3.5/5″2HD版　5,800円(税込)

新作情報

★クイーン・オブ・デュエリスト外伝＋α　TAKERU　8/未
X68000用　3.5/5″2HD版　5,800円(税込)
★魔法大作戦　EAビクター
X68000用　5″2HD版　価格未定
★X CASE　Béシステム
X68000用　5″2HD版　19,800円(税込)
★ロボスポーツ　イマジニア
X68000用　5″2HD版　価格未定
★Traüm　象スタジオ
X68000用　5″2HD版　価格未定
★鮫！　鮫！　鮫！　KANEKO
X68000用　5″2HD版　価格未定
★達人　KANEKO
X68000用　5″2HD版　価格未定
★エアバスター　KANEKO
X68000用　5″2HD版　価格未定
★サバッシュII　ポプコムソフト/グローディア
X68000用　5″2HD版　価格未定
★麻雀悟空・天竺への道　シャノアール
X68000用　5″2HD版　9,800円(税別)
★スタークルーザーII　アルシスソフトウェア
X68000用　5″2HD版　価格未定
★地球防衛MIRACLE FORCE　カスタム
X68000用　5″2HD版　価格未定
★XDTP SX-68K　シャープ　6/未
X68000用　3.5/5″2HD版　価格未定
★スーパーストリートファイターII　カプコン
X68000用　5″2HD版　価格未定
★プリンセスメーカー　ニュー
X68000用　5″2HD版　価格未定
★VIEW POINT　ネクサスインターラクト
X68000用　5″2HD版　価格未定

踊るピエロにドラミファDo!

Sudo Yoshimasa
須藤 芳政

「ドレミファソレシド〜」のリズムに乗って，「Mr.Do!」がビデオゲーム・アンソロジーシリーズに帰ってきた！ さらに，続編の「Mr.Do! vs UNICORNS」も遊べちゃうんだから，これはやっぱりお買い得ってものでしょう。

16年ほど前の私は，名詞の前へ「ザ・」を置いたり，語尾に「〜マン」を付け加えることは最高にカッコイイことだと信じ，学校のトイレで大きいほうを使用することは最高の恥とするごく普通の小学生でした。学校で使うジャ○ニカ学習帳の表紙を「ザ・国語」にしてみたり，ときには「柔道マン」などと名乗り，柔道では使うはずもないパンチやキックを放った拍子に，よろけて本棚のガラスに頭を突っ込んだりしたこともあります（これが原因で頭がおかしくなったとの説あり）。いま考えると，脂汗が吹き出るほどこっ恥ずかしい行為に目を輝かせていたものです。

それから4年後といえば小学6年生。そのころは中学生の「ツッパリ君（死語）」がデパートのゲームコーナーに多く生息。訪れたお子様をトイレへ招待して「お小遣い」をもらうのが流行だったため，ついに学校側から「ゲームセンターへ行ってはいけませーん！」との命が下ってしまいました。

「Mr.Do!」が世に送り出されたのは1982年ですからちょうどこの時期に当たりますね。私は現在までに「Mr.Do!」の名を耳にしてきましたが，プレイはおろか画面すら見たことがありませんでした。全部「ツッパリ君」のせいです。しかし，12年の時を超えビデオゲーム・アンソロジーシリーズの第10弾として御対面となったのです！

2匹揃って天国へお行き！

2本セット?!

今回収録されているのは「Mr.Do!」だけではなく，「Mr.Do!」の続編「Mr.Do! vs UNICORNS」とのセットになっています。これはまさに「カメラを買ったらパノラマカメラがついてくる状態」でしょう。ただし，続編といっても主人公がMr.Doというだけで2つのゲーム内容はまったく異なるので，十分楽しめるお買い得ソフトです。

さて，主人公のMr.Do。私の見るかぎりではピエロを職業としているようです。ピエロという立派な職をもっているにもかかわらず，命を危険にさらして怪物を殺害したり地下に埋もれた土臭いサクランボを食べることが，彼にとってどのようなメリットをもたらすというのでしょうか？ これは彼のプライベートな時間を利用した趣味なのか？ 周囲の人は彼にこの危険な趣味を止めるよう説得してください。

CDEFGAB〜Mr.Do!

「Mr.Do!」は地面をサクサク掘り進みサクランボを食べるゲームです。

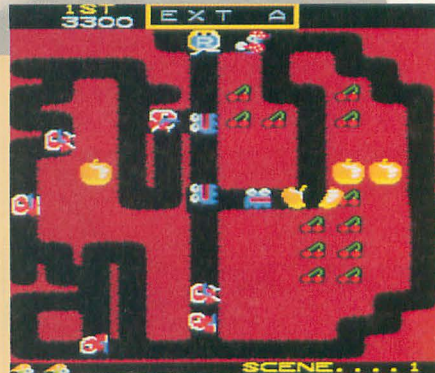
Do君はボールを1個持っています。壁にバウンドを繰り返しながら飛ぶこのボールを，画面中心からワラワラ湧いてくる怪獣にぶつけると怪獣をやっつけられるのですが，その瞬間ボールはバラバラに弾け飛んでしまい，再び破片が元どおりになるまでDo君はボールを投げられません。

で，運よく怪獣を全滅させることができればステージクリアになります。しかし，回を重ねるごとにボールが元どおりになる時間は長くなるようなので，ボールのみで怪獣の相手をするのは非常にデンジャ〜です。怪獣に捕まると「ぴりっぽろっぱらっどぅるん！」というマヌケな音とともにDo君は天へ召されてしまいます。

また，怪獣を倒すにはリンゴを落下させて怪獣を潰す方法もあるので，こちらもうまく利用することにしましょう。

リンゴはDo君よりも大きく，色褪せていて見るからにまずそうです。Do君がリンゴのすぐ下を掘ると，リンゴはグラグラっと揺らいで落下し，押すとズリズリとずらせます。落ちてきたリンゴに当たると，怪獣

X68000用　5"2HD　5,400円(税別)
電波新聞社　☎03(3445)6111

ダイヤは取るもので食べちゃだめ

Rはもらった！

はプチっと潰れて死んでしまいます。もちろんDo君もサイボーグではないので，落ちてくるリンゴには勝てず同様に潰れます。高いところから落ちるとリンゴは「ぶばあ！」と音をたてて割れてしまうので，台風の被害にあったリンゴ園関係者の方は心臓に悪いためプレイを控えましょう。

どうしても怪獣を全滅させられない場合は地面に埋まっているサクランボをすべて食べ尽くしてステージクリアを狙ったほうが無難です。土に埋もれたサクランボをパクパク食べてしまうなんてDo君は野蛮人ですね。友達失っちゃうぞ！

さらに，ごくまれにリンゴの割れた場所からダイヤが出現。このダイヤを取るとステージクリア＆1クレジット追加という豪華プレゼントです。とはいうものの，クレジットはX68000のキーボードを連打すれば好きなだけ投入できるので，1クレジット増えたところでちーっともうれしくありません。だからといって「あの～，1クレジット増えたので100円ください」などと電波新聞社さんを訪れると，おまわりさんに連れていかれるので絶対にやらないように。

文字の描かれた敵を倒して「EXTRA」の5文字を揃えるとDo君が微笑を浮かべ，怪獣をジワジワとしばいて泣かす光景が表示されてDo君が1人追加＋ステージクリア。

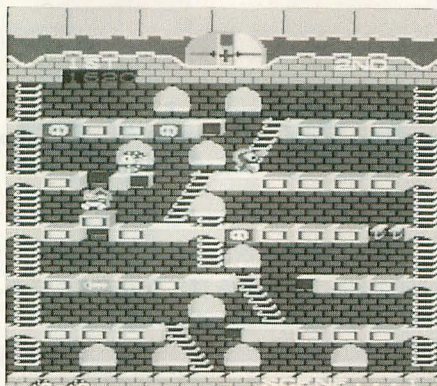
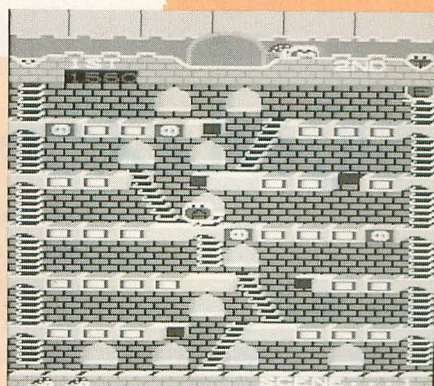
最後に，ひと言。「レバーガチャガチャ」「最上段からリンゴで怪獣と心中」など，発売前から噂になっている「256匹増殖」という極悪技。手元のサンプルバージョンではできないようです。

VS ユニコーンズ!

「ぎーんぎゃろろおがーん！」

これが「Mr.Do! vs UNICORNS」の死亡音。前作の「ぴりっぽろっぱらっどぅるん！」よりも苦痛に満ち溢れたサウンドに，鼓膜がピクピクしてしまいます。

敵は角が1本ピョンと生えたユニコーン。すべてのユニコーンを退治すればステージクリア！

もう敵がうじゃうじゃでいやあ！

さらばユニコーン，君のことは忘れない

オラオラと大量殺戮に走るDo君

で，Do君の今回の武器はトンカチ。このトンカチはひと振りで相手を即死状態へもっていけるほど恐ろしく巨大な凶器です。試しにユニコーンを殴ってみましょう。

「こんにゃろ！ボカ！ボカ！ボカ！」

「ぎーんぎゃろろおがーん！」

驚きです。ユニコーンは殴っても一瞬はひるむのですがすぐに突っ込んできて，結局こちらがやられてしまいました。

どうもトンカチは床にはめ込まれたブロックを叩き落とすために使うようです。このブロックをユニコーンの頭上へ落下させて圧死させるのです。前作のリンゴは自由落下だったのに対して，こちらは思いっきり叩き落とすため非常に残酷な手口といえます。将来動物王国を作ろうと志す方には目を覆いたくなるような光景ですね。

しかし1個ずつチマチマとブロックを投下していてもらちがあきません。なぜなら時間がある程度たつとユニコーンは無限増殖＆超高速化してしまうからです。この状態になってしまったら，手を空気との摩擦で炎が発生するほど速く動かしてスティック操作をしたとしても無理です（ティーンエイジャーにはわかるまい！）。

効率よく相手を倒すにはドクロの描かれたブロックが必要です。このブロックは2つでひと組みになっており，両端のドクロブロックを叩き落とすと2つのブロックの間にあるすべてのブロックが落下します。しかし，そう都合よくその下へユニコーンはきてくれないので，あらかじめ片方のドクロと間にあるブロックを下へ落として空白を作っておきます。するとノコノコやってきたユニコーンがその空白にスポ！っとハマってもがきだすので，すかさず残ったドクロを叩き落とすとユニコーンはズドーンと落下して死んでしまいます。これはもがいている仲間の上を乗り越えてくるユニコーンに対しても有効なので，3つ程度の空白で5匹まとめて病院送り（やはり死んじゃうのかな？）が可能な面もあります。

そして，鍵の描かれたブロックをすべて叩き落とせば最上段にある扉がオープン！そこへDo君がトコトコ歩いてゆくと敵が「EXTRA」の文字に化けるので片っ端から殴っていきましょう。すべて集めればDo君が1人追加。敵が文字に化けている間は接触しても平気なのでペタペタ手垢のつくほど触りまくってくださいね。

どういうわけだか熱い!

さあ，これを死ぬほど練習して「Mr.Do!」のある温泉旅館へゴー！　キミのプレイを見た旅館の主人が，

「ほほお，なかなかのスティックさばきだ！　よし，宿代タダ＆ハワイ旅行だ！」

「Oh！らっきぃー！」

ってそんなに人生あもうないわー！

踊るピエロ防止対策

ディスク1枚のみで，ドライブ0で起動すれば「Mr.Do!」，ドライブ1で起動すれば「Mr.Do! vs UNICORNS」が立ち上がります。

実は私，このゲームはキーボードでプレイしました。いつも使っている8方向スティックだと，斜めに入ったりしてハシゴの上で踊ってしまうなどの奇怪な行動をしてしまうのです（要するに入力ミスが多くなる）。やっぱり，こういうゲームは，4方向スティックで遊びたいものですね。

あと，「Mr.Do! vs UNICORNS」の音はヘッドフォンで聴かないほうがよいでしょう。ピロピロという音が耳にあまりいい感触を与えません。懐かしい雰囲気は出ているんだけどね。

総合評価	0 ～ 5 ～ 10
ピロピロ音	★★★★★★★★★
8方向スティック	★★★★★★
動き	★★★★★★★★
熱中度	★★★★★★★★

お好みレスラーを育てまショー

「レッスルエンジェルス」の第3弾の登場です。これまでの作品はレスラー個人に焦点が当てられていましたが，今回はプロレス団体を結成して選手を育てるのです。さあ，自分好みの選手を集めて，最強団体を作りましょう。

Kiyose Eisuke
清瀬 栄介

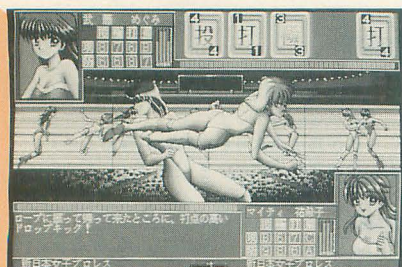
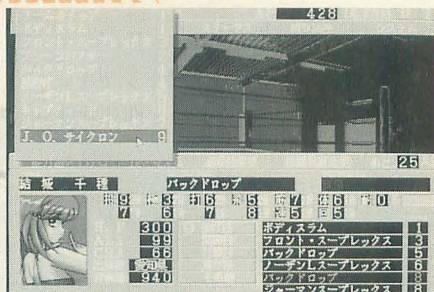
みなさんゴメンなさい。本誌4月号「レッスルエンジェルス2」のレビューで「プログラムと演出は合格点以下」と書いたにもかかわらず，その後3カ月間このゲームだけを遊びまくり，エンディングまで見てしまったのはこのボクです。いやいや，久々に未完成ながらパワーのあるゲームというのに出合ったって感じ。

何が面白いって，ただキャラクターを育てるのが面白い。昔の「ウィザードリィ」のように，プレイに自由度があるとゲームにこだわりが入る余地ができるんだよね。「武藤めぐみのフライングニールキックにかなうものはない」とか，「南利美のSTFはあなどれん」とか。

こだわりがあると，ゲームの世界を自分の思うようにしようとして，展開に執着するようになる。「関節技の防御力をつけて南利美に勝つまでは寝られん」と思ったら，こんどは空中技で競り負けたりして「新しい空中技を覚えるまで寝られん！」。

その目標が達成できた頃には，ビジュアルシーンが入って話が先へ進んでしまう。このリズムの巧みさにボクは睡眠不足のゾンビ状態になってしまったのだった。

社長の趣味はプロデュース

この「レッスルエンジェルス3」では，プレイヤーはプロレス団体の社長として経営を行い，興業の手配やキャラクターの成長の面倒をみる。おぉ，プリンセスメーカ

X68000用　3.5/5"2HD版　5,800円(税込)
TAKERU　☎052(824)2493

なぜか出身地まであるジムのデータ画面

ーならぬ，プロレスラーメーカー。

だがこの変更，実は前作を遊んでいた人にはとても嬉しい変更なのだ。

前作でもどかしかったのが，自分のペースに合わせて周囲の選手が成長してくれないこと。同期でライバルという設定のキャラクターが，後半のほうではどんどん弱くなってしまい，情けない思いをすることがあった。またビジュアルシーンでは対等の立場でかっこつけてるのに，タッグを組むとまるで使えなかったりする。対戦カードはコンピュータが決めるから「こんな弱い奴と交代なんかできないよー」ということで，結局1対2の変則タッグマッチになってしまうことも多かったのだった。

これが今度は団体の社長となって各選手の管理ができるわけだから，選手をどういうふうに売り出すかは自由自在。こいつとこいつをライバルにしてとか，こいつは技の多いテクニシャンにしようとか，さらに海外修行に出したり，覆面をかぶせちゃうこともできる。これで「2」での不満は解消，ハマり度倍増だ。キャラクターと舞台設定は前作と共通なので，キャラクターの性格を知っている人が遊べば自分のこだわりどおりにレスラー陣を構成することができる。ううう，うれしいよーん！

必殺技はレディのたしなみ

「レッスルエンジェルス」の世界では，新日本女子プロレスという団体が幅を利かせている。その頂点に立つのがマイティ裕希子，IWWFアジアヘビー級選手権のチャンピオンである。彼女を凌ぐ選手を育てるのが社長であるプレイヤーの目標だ。

まずは選手を集めてくる。最低でも6人の選手がいないと旗揚げができない。集める方法には，新日本女子から引き抜くか，引退した選手や無所属の選手をスカウトするか，新人を発掘するという3つがある。やはりここはボクの好みをいわせてもらうとだね，前作の主人公，武藤めぐみと結城千種のどちらかは当然引き抜きたいね。人気があるわりに移籍金が安いキューティ金井もお薦め。あとは引退選手のなかからブレード上原，サンダー龍子といった実力派をスカウトし，新人を採用すれば布陣は整うはずだ。

6人揃ったら，全国地図を前に興業を行う場所を選ぶ。会場名が大田区体育館やら北上家畜市場特設リングやら，妙にディテールが細かいのがおかしい。自分の団体の人気と相談のうえで，場所と回数を決めよう。興業が長すぎると選手がケガをしやすくなるので，そのへんも見極めたい。

次は対戦カードを決める。各レスラーには評価値という総合的な実力を表すパラメータがあるので，それを参考にしよう。メインイベントは評価の高い選手が出たほうがお客さんが来る。また評価値の低い選手が高い選手に勝つと，選手の人気が上がる。

便利なマッチメイクのやり方は，評価値の高い選手を左に，低いほうを右に置く方法だ。格上の選手から金星を取ったかどう

お嬢さん，ボクとマットで格闘しない？

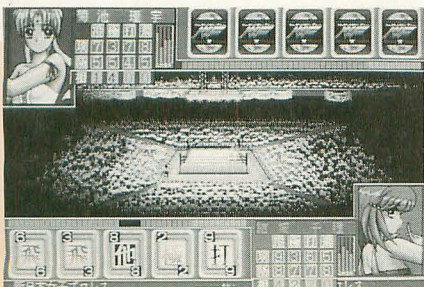
かがわかりやすい。プロレスの世界では，チャンピオンが左側にくるのがお約束ということでもあるしね。

いよいよ試合だ。結果だけ見てもいいが，なんといってもこれがゲームのヤマでありキモであるからして，お気に入りの選手の試合や大事な一戦のときは，自分の手でプレイするべきだろう。

画面はおなじみのカードバトル方式。といっても，完全なカードゲームではなくて，普段から鍛えている能力が勝負を大きく左右する。基本的には双方の選手が投げ技，空中技などのカードを切り，そのなかでもどの技を出したいかを指定する。それを計算して，どちらの攻撃が勝ったかを決めるという仕組みだ。

さぁ結城千種対サンダー龍子，先ほどからアームホイップやエルボーが飛び交っています。やや小技の応酬に終始しているか。あっと，サンダーのラリアートがのど元にヒット！　負けじと結城もフロントスープレックスで鋭く投げ返す。……相手がどんなカードを出してくるかを読みながら技を選ぶのが勝負の駆け引きである。

突然オールマイティのカードが切られた。必殺技サンダーボム！　負けじとこちらも取っておきの投カードでジャーマンスープレックス。おおっと，サンダー龍子がパワースラムでフォールにいった。カウント1，2，2.5で返す！　千種も負けじとラリアート……あー，読まれて逆ラリアートだぁ！千種，体力が尽きた！　もうダメか，カウント1，2，スリ……返した，返した，カウント2.9で返した！　さぁ，だが結城千種ピンチ。あー！　ここでついに出たかオールマイティのカード！　必殺のジャパニーズオーシャンサイクロンスープレックスホールドぉ(長い)!!　難易度Dの幻の必殺技，J.O.サイクロンだー！　カウントが入る，1，2，……3，決まったぁー！　10分23秒J.O.サイクロンで結城千種が勝ちましたぁー！

大事な一戦をプレイしていると，これくらいのテンションの高さが味わえる。特に

試合では敵の手札の読みが重要だ

写真集ではレスラーの意外な顔が見られるぞ

団体対抗戦ではこんなカードも実現

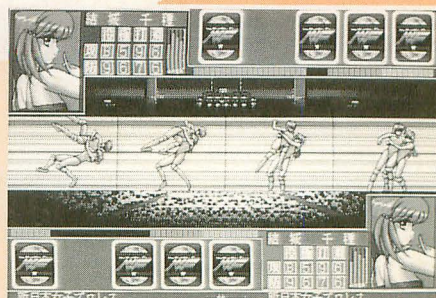
この「3」では，カウント2から3までがやけに長かったり，体力が尽きてもなかなかフォールされなかったりなどの演出がされているせいで，前作より盛り上がるぞ。

試合の結果に応じて，レスラーには経験値が与えられる。これでトレーニングをして能力を高めたり，新しい技を覚えたり，必殺技を変えたりするわけだ。

必殺技は普通のカードで出すこともできる。が，ある技を必殺技として習得すると，まずオールマイティのカードが出たときにその技を無条件でかけることができる。

そして，必殺技には好きな名前がつけられる。これが単純だが面白い。自分の好きなレスラーの必殺技に名前をつける。バカバカしいと知りつつも，試合中にナレーションがその名前を叫んでくれると嬉しくなってしまう。最初はテレながら名前を考えていたボクだが，いまじゃテレもなく「よっしゃ，レインボーキャプチュード炸裂！」とかいって，モニターの前ではしゃいでいる。作った人はプロレスもよくわかってるし，ゲーム作りもよくわかっているようだ。

B級パワーを信じて買え

ボクはいま「私の睡眠時間を返せ」モードに入っている。結城千種とサンダー龍子の2人はいまや世界に誇るレスラーに育った。あとはIWWFヘビー級王座を取るだけだ。

とまあ，個人的には非常にハマっているのだが，これが他人にも薦めやすいかどうかとなるとちと問題がなくもない。見た目がかなりB級ソフトなので，一生懸命ボクが薦めれば薦めるほどウサン臭くなるという宿命があるのである。

ひいき目に見ても，アートディレクションはもうちょっとなんとかならんかなという気がする。各画面の色調をもっと揃えてもらいたい。特に試合の画面は色も文字もゴチャゴチャしていてセンスが悪い。

それからキャラクターのタッチ。まず選手ごとのグラフィックが揃ってないし，同じ選手でも通常画面と写真集を出したときで全然違ったりする。複数のグラフィッカーが好き勝手に描いていては，品質が高いソフトにはどうしても見えない。それに，技の決めグラフィックは，前作のほうがよっぽどキレイで動きがあったぞ。

操作性，プログラムは前作より進化が著しく，ほとんど気にならないレベルまできている。10MHz＋FDの最低ランクの環境でも，速度に不満はない。

PC-9801版では次作「スペシャル」もあるので，X68000版もボクはすごーく期待大。もう女子プロになじみがないとか，プロレスのことはよくわからないとか，そういうことは関係ない。これを遊んでプロレスを学ぶのだ。このシリーズ，うまく育てば相当ビッグネームになれるかもよ。

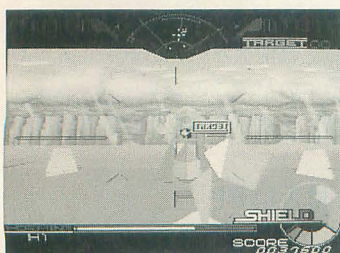
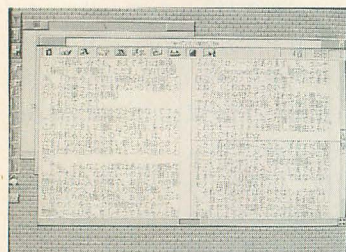
トップイベンターは近いぞ!

「2」はプログラムも演出もプアながら，どこかハマる要素をもったゲームだった。この「3」は，ハマる要素をいっそう強くしながら，前作の弱かったところをかなり補強している。プレイヤーのこだわりが入り込めるようにした細かな部分や，フォールのときの演出などがそれだ。前作のレスラー個人から団体社長へと，プレイヤーの立場は変わったが，ゲームの面白さの性質はほとんど変わっていないのが嬉しい。

次作「レッスルエンジェルススペシャル」は再びキャラクターを操るゲームに戻るということだが，「3」から団体経営ゲームとして発展させる路線を別に作るのも面白いかもしれない。現時点ではあまり興業成績に工夫の入る余地はないが，これがマネジメントゲームとして遊べても楽しいだろうという気はする。

とにかく最近遊んだなかではいちばん先行きが楽しみなシリーズだ。

総合評価	0　5　10
操作性	★★★★★★★★
音楽	★★★★★
グラフィック	★★★★★★
ゲーム性	★★★★★★★★★
熱中度	★★★★★★★★★

TREND ANALYSIS

1994年7月号のハガキ集計ベスト10 最近買って気に入ったソフトは？

POINT	タイトル	発売元	発売日
84	SX-WINDOW ver.3.1	シャープ	'94/5/30
60	ジオグラフシール	エグザクト	'94/3/15
45	ぷよぷよ	SPS	'94/3/25
33	あすか120% BURNING Fest.	ファミリーソフト	'94/4/22
25	スーパーリアル麻雀PⅣ	ビング	'94/4/27
21	アルゴスの戦士	電波新聞社	'94/4/27
21	大魔界村	カプコン	'94/4/22
17	餓狼伝説2	魔法株式会社	'93/12/23
12	悪魔城ドラキュラ	コナミ	'93/7/23
10	ストリートファイターⅡダッシュ	カプコン	'93/11/26

（無作為抽出した1000通のハガキを集計）

先月号の「期待のソフト」の1位に挙がっていた「SX-WINDOW ver.3.1」ですが，期待を裏切らぬ内容だったようで，さっそく1位の座を奪い取りました。発売が開始されたのは5月30日で，この集計の時点ではまだ1カ月も経過していません。発売を待っていて即，購入し，この読者アンケートはがきを書いた，という人たちによる獲得ポイントです。しかし，そういう人ばかりではなく，22ページの「期待のソフト」の集計でも5位となっていますので，今後購入を予定している人もたくさんいるようです。来月号ではさらに獲得ポイントが伸びる可能性もあります。

ゲーム以外のソフトがトップに立ったのは，この読者はがきによる集計を始めて以来，初の出来事です。評判のよかった前回の「ver.3.0」でも40ポイント獲得で2位というのが最高だったことを考えると，ウィンドウシステムの浸透度と，今後への期待が加速されつつあるのが感じられるようです。なかには，「ver.1」や「ver.2」からバージョンアップしたという人も数人いて，「すごく進化していて驚いた」というコメントがありました。

製品紹介は70ページで行っていますので，参考にしてくださいね。

2位から6位までは3月と4月に発売のゲームが並びました。どれもそろそろ遊び込んだ頃なのでしょうか。それぞれ，ジャンルや雰囲気がいろいろなゲームですので，好みによって評価が分かれたようです。しかし，その分，思い入れ度が高そうだな，ということがコメントなどからもうかがわれます。

8位から10位には，1993年に出た人気のビッグタイトルが，発売から数カ月たっているにもかかわらず，まだまだがんばっています。特に「悪魔城ドラキュラ」は1993年9月号が初登場でしたから，もう1年間もランキング入りを続けているわけです。まあ，GAME OF THE YEARにて圧倒的人気で受賞を果たした作品ですから，驚くことではないのかもしれません。「餓狼伝説2」と「ストリートファイターIIダッシュ」については，それぞれ次回作の発売が予定されていますので，そろそろ「政権交代」といったところでしょうか。

さて，来月以降について考えてみると，トップポイント獲得の大本命「餓狼伝説SPECIAL」は7月28日発売ですので，来月号の集計には間に合わないと思われます。しかし，期待が高いゲームであるだけに，さ来月あたりでどういう結果が出るかが楽しみです。来月は，最初に述べたように「SX-WINDOW ver.3.1」がポイントを増やして1位を守るか，それともほかのソフトが奪い取るか。どうなるでしょうか。

Graphic Movement

[特集]
グラフィックムーブメント

グラフィックパワー
いまや中途半端といわれるX68000の表示能力
これまでアンチエイリアスで解像度を誤差拡散で色数を
アルゴリズムで処理速度を補ってきた
X68000のグラフィックパワーはいまだトップレベルだ
しかしハードウェア的なブレイクスルーが求められているのも事実
X68000のグラフィックはひとつの転機を迎えつつある
そしていま，SX-WINDOWが標準プラットホームとして注目されてきている
SX-WINDOW上でどのような環境を確立していくかを
本格的に考えねばならないときがきたのだ
いまだ混沌としたSX-WINDOWのグラフィック環境
X68000のグラフィックパワーはウィンドウ上で眠りについている
アプリケーションの連係を十分に考え
ハードウェアの機能を存分に生かす道を探ってみよう

CONTENTS

2D画像フィルタを作る

EX-WINDOW用アクセサリ

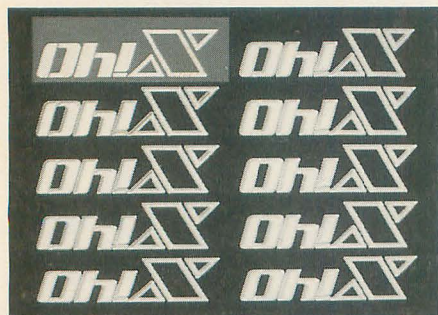
Kikuchi Isao 菊地　功

まずは基本のペイント系2Dグラフィックから。EX-WINDOWを使えば必要な機能はいくらでも拡張できます。今回は文字などをガラス状にするフィルタと色チャネルごとの処理を可能にするプレーン分離フィルタです。

ちょっと前まではグラフィックでは大きくほかのマシンに水を開けていたX68000ですが，最近は国民機や互換機のグラフィックアクセラレータの普及で，ちょっと寂しいかなって気もする今日この頃です。私も今年の2月に互換機をバラで組み，S3という会社の86C928というグラフィックチップを載せました。この928という石，かなりシェアがあり結構な速さなんですけど，ついこのあいだ故障してしまい，出たばかりの同社の86C964という石の載ったボードに差し替えました(ちょっと自慢)。これが速いのなんのって，486DX2-66ではもったいないくらいです。別にそんなに速くなくてもいいからX68000でもグラフィックアクセラレータがどこかから出ませんかね。そしたらSXももっと快適になるでしょうに。まあ，X68000の場合は標準で512KバイトのVRAMを積んでいるのが強みでもあるんですが。

ということで今年の3～5月号で発表したEX-WINDOW専用の外部ファイルです。EX-CALLを使っているので，それ以前のZ's-EXでは動作しません。あしからず。

ガラスのような質感のロゴを作る

まず1本目，GLASS.Xです。ちょっと盛り上がったガラスのような質感のロゴ作成用の外部ファイルです(リスト1)。やっていることは，適度にぼかして，ちょっとずらしてから輝度の差を取って，縁を切り出すだけなんですが，これを1パスでやるのがミソです。そのため，ロゴ作り以外に用途はないかもしれませんが。

ガラスロゴ

shade()という関数でぼかしをかけていますが，単純平滑化をかける回数でぼかしのレベルを変えています。本来ならガウスぼかしなどのすべきかもしれませんが，あまり結果が変わるとも思えなかったので簡単に済ませてしまいました。また，最後の輪郭の切り出しには1回目のぼかしのエッジを使いたかったため，1回目のループだけ独立させ待避するようにしています。こうしておけばエッジの情報を持たせるためのメモリを別に確保してやる必要がなくなります。

そのあとでsub()という関数で2ドット左上のピクセルとの差の絶対値を取っています。この関数では矩形範囲の左上のピクセルを背景とみなして縁を切り出し，周囲を黒での塗りつぶしも行っています。

リスト1をglass.cというファイル名で入力したらコンパイルしてみましょう。EXライブラリをリンクさせることを忘れないでください（3月号付録ディスクに収録)。GCCならば，

```
gcc -O glass.c -ldos -lfloatfnc -lex
```

XCならば，

```
cc glass.c exlib.l /Y
```

でいいでしょう。ただし，ここで使うライブラリはXC付属のライブラリですので，LIBCなどのライブラリではそのままコンパイルできないかもしれません（私は持っていないので)。

コンパイルができたら，生成されたglass.xをパスの通ったディレクトリに移してコンフィグファイルに次のように記述してください（コンフィグファイルについては3月号参照のこと)。

```
:ガラスロゴ
	1, 1: 1-9, 4
	glass.x
```

システム側で矩形指定を行いますが，矩形指定フラグが0の時は画面全体が対象となります。パラメータは1個で，ぼかしの度合を示します。省略時は4となります。

では実際に使用してみましょう。EX-WINDOWからガラスロゴを指定すると，まず矩形指定モードになります。マウスの左ボタンで対象となる領域を指定するのですが，このとき処理を行う領域より大きめに，なおかつ矩形の左上が背景の位置になるように指定してください。うまくいけば写真1のようなロゴが得られるはずです。なお，マスクは無視されますので，注意してください。パラメータを変えたり，背景と文字の色の組み合わせを変えることで，少しずつ違った質感になりますので気にいったロゴを見つけてください。

プレーンの抽出/合成をする

続いては，任意のプレーンの抽出/合成を行うPLANE.Xです。指定できるプレーンはお馴染みのRGBと，色相，彩度，明度からなるHSVです。このうちRGBとSVは0～31で表されますので，32段階のグレースケールで表示できますが，Hだけは192段階でそうもいきませんでしたので，SとVをともに31 (100%) とした場合の色で表現することにしました。

抽出は裏画面から任意のプレーンを表画面に，合成は裏画面を任意のプレーンとして表画面に転送するという方法にしました。抽出と合成で元画像とプレーンが逆ですが，裏画面を直接更新したくなかったからです。更新するのは常に表画面と覚えておいてください。

では，さっそくコンパイルしてみましょう。プログラムは5月号で発表された吉田

泉氏のデザイナでスケルトンを作って手を加えていきました。したがって，このプログラムをコンパイルするにはデザイナのライブラリ（deslib.c）が必要です（EXライブラリに結合している方は必要ありません）。また，deslib.hを同じディレクトリに置くか，環境変数includeで指定したディレクトリに置いてある場合にはdeslib.cとplane.cの，

　　#include　　　"deslib.h"

の行を，

　　#include　　　＜deslib.h＞

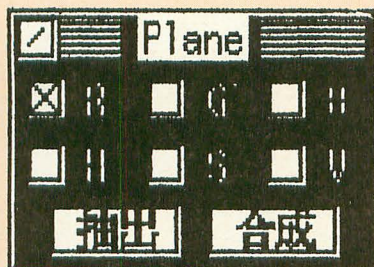
に変更してください。

コンパイルオプションですが，deslib.cが*BASIC*ライブラリを使用していますので，XCでコンパイルするのなら，

　　cc plane.c deslib.c exlib.l /Y /W

としてください。さてgccですが，次のようにしてコンパイルしてください。

　　gcc -O -fomit-frame-pointer plane.c deslib.c -ldos -lfloatfnc -lbas -lex

ここで，-fomit-frame-pointerというオプションですが，まとめてスタックを解放する最適化を抑止するものです。GCCのバージョンにもよりますが，スーパーバイザモードとユーザーモードを行き来するプログラムではこのオプションをつけないとユーザースタックポインタとスーパースタックポインタに矛盾が生じてしまうからです。まあ，そういうものなんだと覚えておきましょう。

無事にコンパイルが済んだら，コンフィグファイルに次のように記述しましょう。

：プレーン

　　0，0

　　plane.x

要するにフラグもパラメータもなしで起動させるわけですね。

さて，EXからプレーンを選択すると図1のようなウィンドウが開きます。チェックボックスで任意のプレーンを指定して，抽出または合成のボタンで実行してください。先ほどの説明の通り裏画面から任意のプレーンを表画面に抽出，または裏画面を任意のプレーンとして表画面に合成されます。これがなんの役に立つのかと聞かれるとちょっと困ってしまうのですが，特定のプレーンにいろいろなエフェクトをかけることで，面白い効果が得られるかもしれませんよ。お試しあれ。

図1

輝度のみを抽出してみたところ

飽和度を抽出するとこうなる

ネタ募集

そろそろ外部ファイルのネタも切れてきた感じです（まだいくつかはありますが，ちょっと紙面に載せるのは厳しいものばかりなので）。そこで読者の方々からネタを募集します。こんなのあったらいいな，こんなことできたら便利だなというものがありましたら，編集部に送ってください。できそうなものであれば，採用したいと思います。賞品はありませんけどね。もちろんプログラムもお待ちしています。

EX-WINDOWの本体もかなり進化しています。次回の発表時には（まだいつかわかりませんが）またひと味違ったEXを提供できるかと思いますので，お楽しみに。では，また。

リスト1

```
 1: #include     <stdlib.h>
 2: #include     <stdio.h>
 3: #include     <doslib.h>
 4: #include     <iocslib.h>
 5: #include     <exlib.h>
 6:
 7: #define      Set(x,y)          ¥
 8:     {¥
 9:              g = (((x)>>11)&0x1F)<<(y);¥
10:              r = (((x)>> 6)&0x1F)<<(y);¥
11:              b = (((x)>> 1)&0x1F)<<(y);¥
12:     }
13:
14: #define      Add(x,y)          ¥
15:     {¥
16:              g += (((x)>>11)&0x1F)<<(y);¥
17:              r += (((x)>> 6)&0x1F)<<(y);¥
18:              b += (((x)>> 1)&0x1F)<<(y);¥
19:     }
20:
21: #define      Sub(x)  ¥
22:     {¥
23:              g -= ((x)>>11)&0x1F;¥
24:              r -= ((x)>> 6)&0x1F;¥
25:              b -= ((x)>> 1)&0x1F;¥
26:              if( g<0 ) g = -g;¥
27:              if( r<0 ) r = -r;¥
28:              if( b<0 ) b = -b;¥
29:     }
30:
31: void         shade();
32: void         sub();
33:
34: int          main( int ac, char *av[] )
35: {
36:     int      i=4;
37:     int      x1=0, y1=0, x2=511, y2=511;
38:     unsigned short  *buf, *vp;
39:
40:     if( ac>=6 ){
41:              x1 = atoi( av[2] );
42:              y1 = atoi( av[3] );
43:              x2 = atoi( av[4] );
44:              y2 = atoi( av[5] );
45:              if( ac>=7 ) i = atoi( av[6] );
46:     } else  if( ac>=3 ) i = atoi( av[2] );
47:     vp = (unsigned short *)0xC00000;
48:     vp += x1+y1*512;
49:     buf = BUFFADR();
50:     buf += x1+y1*512;
51:     SUPER( 0 );
52:     shade( vp, buf, x2-x1, y2-y1, i );
53:     sub( vp, buf, x2-x1, y2-y1 );
54:     return( 0 );
55: }
56:
57: unsigned short      lbuf[2][512];
58:
59: void         shade( unsigned short *vp, unsigned short *buf, int dx, int dy,
int i )
60: {
61:     int      x, y;
62:     int      r, g, b;
63:     int      l0, l1;
64:
65:     for( x=0; x<=dx; x++ ) lbuf[0][x] = buf[x];
66:     for( y=1; y<dy; y++ ){
67:              l0= y%2;
68:              l1 = (y-1)%2;
69:              for( x=0; x<=dx; x++ ) lbuf[l0][x] = buf[x+y*512];
70:              for( x=1; x<dx; x++ ){
71:                       Set( lbuf[l0][x], 2 );
72:                       Add( lbuf[l0][x-1], 1 );
73:                       Add( lbuf[l0][x+1], 1 );
74:                       Add( lbuf[l1][x], 1 );
75:                       Add( buf[x+(y+1)*512], 1 );
76:                       Add( lbuf[l1][x-1], 0 );
77:                       Add( lbuf[l1][x+1], 0 );
78:                       Add( buf[x-1+(y+1)*512], 0 );
79:                       Add( buf[x+1+(y+1)*512], 0 );
80:                       buf[x+y*512] = vp[x+y*512] = (((g+8)<<7)&0xF800)|(((r+8)
<<2)&0x07C0)|(((b+8)>>3)&0x003E);
81:              }
82:     }
83:     for( i--; i>0; i-- ){
84:     for( x=0; x<=dx; x++ ) lbuf[0][x] = vp[x];
```

```
 85:     for( y=1; y<dy; y++ ){
 86:             l0= y%2;
 87:             l1 = (y-1)%2;
 88:             for( x=0; x<=dx; x++ ) lbuf[l0][x] = vp[x+y*512];
 89:             for( x=1; x<dx; x++ ){
 90:                     Set( lbuf[l0][x], 2 );
 91:                     Add( lbuf[l0][x-1], 1 );
 92:                     Add( lbuf[l0][x+1], 1 );
 93:                     Add( lbuf[l1][x], 1 );
 94:                     Add( vp[x+(y+1)*512], 1 );
 95:                     Add( lbuf[l1][x-1], 0 );
 96:                     Add( lbuf[l1][x+1], 0 );
 97:                     Add( vp[x-1+(y+1)*512], 0 );
 98:                     Add( vp[x+1+(y+1)*512], 0 );
 99:                     vp[x+y*512] = (((g+8)<<7)&0xF800)|(((r+8)<<2)&0x07C0)|((
(b+8)>>3)&0x003E);
100:             }
101:     }
102:     }
103: }
104:
105: void        sub( unsigned short *vp, unsigned short *buf, int dx, int dy )
106: {
107:     int     x, y;
108:     int     r, g, b;
109:     unsigned short  col;
110:
111:     col = *buf&0xFFFE;
112:     for( y=dy; y>=0; y-- ){
113:             for( x=dx; x>=0; x-- ){
114:                     if( buf[x+y*512]==col ) vp[x+y*512] = 0;
115:                     else if( x>=2 && y>=2 ){
116:                             Set( vp[(x-2)+(y-2)*512], 0 );
117:                             Sub( vp[x+y*512] );
118:                             vp[x+y*512] = ((g<<11)|(r<<6)|(b<<1))^0xFFFE;
119:                     }
120:             }
121:     }
122: }
```

リスト2

```
  1: /*
  2:  * EX-Window Ver.2.0 X file.
  3:  * (c) i.kikuchi & i.yoshida
  4:  */
  5:
  6: #include    <stdlib.h>
  7: #include    <stdio.h>
  8: #include    <graph.h>
  9: #include    <doslib.h>
 10: #include    <string.h>
 11: #include    <iocslib.h>
 12: #include    <mouse.h>
 13: #include    <exlib.h>
 14:
 15: #include    "deslib.h"
 16:
 17: void        check_box_01( int i );  /* ID  0-5 */
 18: void        button_01( int i );     /* ID  6 */
 19: void        button_02( int i );     /* ID  7 */
 20:
 21: void        open_win();
 22: void        getRGB();
 23: void        putRGB();
 24: void        getH();
 25: void        getS();
 26: void        getV();
 27: void        putH();
 28: void        putS();
 29: void        putV();
 30: void        RGBtoHSV();
 31: void        HSVtoRGB();
 32: int         max3();
 33: int         min3();
 34:
 35: ITEM        item[] = {
 36:     255, 1,   3,   3,  13,  13,   4,   4,  12,  12,
 37:     254, 0,  16,   3, 109,  13,   0,   0,   0,   0,
 38:       0, 1,   4,  18,  17,  31,   5,  19,  16,  30  /* ID  0 */,
 39:       0, 1,  40,  18,  53,  31,  41,  19,  52,  30  /* ID  1 */,
 40:       0, 1,  76,  18,  89,  31,  77,  19,  88,  30  /* ID  2 */,
 41:       0, 1,   4,  37,  17,  50,   5,  38,  16,  49  /* ID  3 */,
 42:       0, 1,  40,  37,  53,  50,  41,  38,  52,  49  /* ID  4 */,
 43:       0, 1,  76,  37,  89,  50,  77,  38,  88,  49  /* ID  5 */,
 44:       1, 1,  13,  56,  50,  69,  14,  57,  49,  68  /* ID  6 */,
 45:       1, 1,  60,  56,  97,  69,  61,  57,  96,  68  /* ID  7 */,
 46:       0, 1,  18,  18,  31,  31,   0,   0,   0,   0  /* ID  8 */,
 47:       0, 1,  54,  18,  67,  31,   0,   0,   0,   0  /* ID  9 */,
 48:       0, 1,  90,  18, 103,  31,   0,   0,   0,   0  /* ID 10 */,
 49:       0, 1,  18,  37,  31,  50,   0,   0,   0,   0  /* ID 11 */,
 50:       0, 1,  54,  37,  67,  50,   0,   0,   0,   0  /* ID 12 */,
 51:       0, 1,  90,  37, 103,  50,   0,   0,   0,   0  /* ID 13 */,
 52:       0, 0,   0,   0, 109,  74,   0,   0,   0,   0  /* ID 14 */,
 53:       0, 0,-511,-511, 511, 511,   0,   0,   0,   0  /* ID 15 */
 54: };
 55:
 56: struct      ct_t
 57: ctrl[] = {
 58:     5,      "",             1, 0   /* ID  0 */,
 59:     5,      "",             0, 0   /* ID  1 */,
 60:     5,      "",             0, 0   /* ID  2 */,
 61:     5,      "",             0, 0   /* ID  3 */,
 62:     5,      "",             0, 0   /* ID  4 */,
 63:     5,      "",             0, 0   /* ID  5 */,
 64:     2,      "抽出",         1, 0   /* ID  6 */,
 65:     2,      "合成",         1, 0   /* ID  7 */,
 66:     8,      "R",           -1, 0   /* ID  8 */,
 67:     8,      "G",           -1, 0   /* ID  9 */,
 68:     8,      "B",           -1, 0   /* ID 10 */,
 69:     8,      "H",           -1, 0   /* ID 11 */,
 70:     8,      "S",           -1, 0   /* ID 12 */,
 71:     8,      "V",           -1, 0   /* ID 13 */
 72: };
 73:
 74: struct      ITEMPTR itemptr = {
 75:     -1, -1, 110, 75, 18
 76: };
 77:
 78: int         PlaneNo = 0;
 79: /* 0:R 1:G 2:B 3:H 4:S 5:V */
 80:
 81: int         main( int argc, char *argv[], char *envp ){
 82:     int     i;
 83:
 84:     open_win();
 85:
 86:     MCSET( 0 );
 87:     for( ; ; ){
 88:             i = MANAGE( 1 );
 89:             switch( i%256 ){
 90:                     case 255:/* ESC */
 91:                             CLOSEWIN();
 92:                             return( -1 );
 93:                     case   0:/* CLOSE */
 94:                             CLOSEWIN();
 95:                             return( 2 );
 96:             /*      case   1:/* MOVE */
 97:                             break;
 98:                     case   2:/* ID  0 */
 99:                     case   3:/* ID  1 */
100:                     case   4:/* ID  2 */
101:                     case   5:/* ID  3 */
102:                     case   6:/* ID  4 */
103:                     case   7:/* ID  5 */
104:             /*              check_box(  0 );*/
105:                             check_box_01( i );
106:                             break;
107:                     case   8:/* ID  6 */
108:                             CLOSEWIN();
109:                             button_01( i );
110:                             open_win();
111:                             break;
112:                     case   9:/* ID  7 */
113:                             CLOSEWIN();
114:                             button_02( i );
115:                             open_win();
116:                             break;
117:                     case 10:/* ID  8 */
118:                     case 11:/* ID  9 */
119:                     case 12:/* ID 10 */
120:                     case 13:/* ID 11 */
121:                     case 14:/* ID 12 */
122:                     case 15:/* ID 13 */
123:                             check_box_01( i-8 );
124:                             break;
125:             }
126:     }
127:     CLOSEWIN();
128:     return( 2 );
129: }
130:
131: void        check_box_01( int i )
132: {
133:     i = i%256-2;     /* chekcbox の no. */
134:     check_box( PlaneNo );
135:     check_box( i );
136:     PlaneNo = i;
137: }
138:
139: void        button_01( int i )         /* 抽出 */
140: {
141:     MCSET( 1 );
142:     switch( PlaneNo ){
143:     case 0: /* R */
144:             getRGB( 6 );
145:             break;
146:     case 1: /* G */
147:             getRGB( 11 );
148:             break;
149:     case 2: /* B */
150:             getRGB( 1 );
151:             break;
152:     case 3: /* H */
153:             getH();
154:             break;
155:     case 4: /* S */
156:             getS();
157:             break;
158:     case 5: /* V */
159:             getV();
160:             break;
161:     }
162:     MCSET( 0 );
163: }
164:
165: void        button_02( int i )         /* 合成 */
166: {
167:     MCSET( 1 );
168:     switch( PlaneNo ){
169:     case 0: /* R */
170:             putRGB( 6 );
171:             break;
172:     case 1: /* G */
173:             putRGB( 11 );
174:             break;
175:     case 2: /* B */
176:             putRGB( 1 );
177:             break;
178:     case 3: /* H */
179:             putH();
180:             break;
181:     case 4: /* S */
182:             putS();
183:             break;
184:     case 5: /* V */
185:             putV();
186:             break;
187:     }
188:     MCSET( 0 );
189: }
190:
191: /*
192:  * EX-Window Ver.2.0 X file.
193:  * End of Sorce File
194:  */
195:
196: void        open_win()
197: {
198:     itemptr.i = item;
199:     set_window_item( &itemptr, &ctrl[0], &item[0] );
200:     OPENWIN( "Plane" );
201:
202:     show_all_ctrl();
203: }
204:
205: void        getRGB( int shift )
206: {
```

▶ 7月号の特集を読んで，「MUSIC SX-68K」が欲しくなった。私の場合，楽譜を読むのには困らないから，ちょうどいいかもしれない。　小笠原　浩一(21)千葉県

```
207:     unsigned short  *gram, *another;
208:     int      x, y;
209:     int      ssp;
210:     unsigned short  level;
211:
212:     gram = (unsigned short *)0xC00000;
213:     another = GETADR();
214:     ssp = SUPER( 0 );
215:     for( y=0; y<512; y++ ){
216:             for( x=0; x<512; x++ ){
217:                     level = (another[x+y*512]>>shift)&0x1F;
218:                     gram[x+y*512] = (level<<11)|(level<<6)|(level<<1);
219:             }
220:     }
221:     SUPER( ssp );
222:     COMGVRAM( 0 );
223: }
224:
225: void        putRGB( int shift )
226: {
227:     unsigned short  *buffer, *gram, *another;
228:     int      x, y;
229:     int      ssp;
230:     unsigned short  mask;
231:
232:     mask = 0x1F<<shift;
233:     buffer = BUFFADR();
234:     gram = (unsigned short *)0xC00000;
235:     another = GETADR();
236:     ssp = SUPER( 0 );
237:     for( y=0; y<512; y++ ){
238:             for( x=0; x<512; x++ ){
239:                     gram[x+y*512] = (buffer[x+y*512]&(mask^0xFFFF))
240:                                     |(another[x+y*512]&mask);
241:             }
242:     }
243:     SUPER( ssp );
244:     COMGVRAM( 0 );
245: }
246:
247: void        getH()
248: {
249:     int      r, g, b, h, s, v;
250:     unsigned short  *gram, *another;
251:     int      x, y;
252:     int      ssp;
253:
254:     gram = (unsigned short *)0xC00000;
255:     another = GETADR();
256:     ssp = SUPER( 0 );
257:     for( y=0; y<512; y++ ){
258:             for( x=0; x<512; x++ ){
259:                     g = another[x+y*512]>>11;
260:                     r = (another[x+y*512]>>6)&0x1F;
261:                     b = (another[x+y*512]>>1)&0x1F;
262:                     RGBtoHSV( r, g, b, &h, &s, &v );
263:                     HSVtoRGB( h, 31, 31, &r, &g, &b );
264:                     gram[x+y*512] = (g<<11)|(r<<6)|(b<<1);
265:             }
266:     }
267:     SUPER( ssp );
268:     COMGVRAM( 0 );
269: }
270:
271: void        getS()
272: {
273:     int      r, g, b, h, s, v;
274:     unsigned short  *gram, *another;
275:     int      x, y;
276:     int      ssp;
277:
278:     gram = (unsigned short *)0xC00000;
279:     another = GETADR();
280:     ssp = SUPER( 0 );
281:     for( y=0; y<512; y++ ){
282:             for( x=0; x<512; x++ ){
283:                     g = another[x+y*512]>>11;
284:                     r = (another[x+y*512]>>6)&0x1F;
285:                     b = (another[x+y*512]>>1)&0x1F;
286:                     RGBtoHSV( r, g, b, &h, &s, &v );
287:                     gram[x+y*512] = (s<<11)|(s<<6)|(s<<1);
288:             }
289:     }
290:     SUPER( ssp );
291:     COMGVRAM( 0 );
292: }
293:
294: void        getV()
295: {
296:     int      r, g, b, h, s, v;
297:     unsigned short  *gram, *another;
298:     int      x, y;
299:     int      ssp;
300:
301:     gram = (unsigned short *)0xC00000;
302:     another = GETADR();
303:     ssp = SUPER( 0 );
304:     for( y=0; y<512; y++ ){
305:             for( x=0; x<512; x++ ){
306:                     g = another[x+y*512]>>11;
307:                     r = (another[x+y*512]>>6)&0x1F;
308:                     b = (another[x+y*512]>>1)&0x1F;
309:                     RGBtoHSV( r, g, b, &h, &s, &v );
310:                     gram[x+y*512] = (v<<11)|(v<<6)|(v<<1);
311:             }
312:     }
313:     SUPER( ssp );
314:     COMGVRAM( 0 );
315: }
316:
317: void        putH()
318: {
319:     int      r, g, b, h, s, v, i;
320:     unsigned short  *buffer, *gram, *another;
321:     int      x, y;
322:     int      ssp;
323:
324:     buffer = BUFFADR();
325:     gram = (unsigned short *)0xC00000;
326:     another = GETADR();
327:     ssp = SUPER( 0 );
328:     for( y=0; y<512; y++ ){
329:             for( x=0; x<512; x++ ){
330:                     g = another[x+y*512]>>11;
331:                     r = (another[x+y*512]>>6)&0x1F;
332:                     b = (another[x+y*512]>>1)&0x1F;
333:                     RGBtoHSV( r, g, b, &h, &s, &v );
334:                     g = buffer[x+y*512]>>11;
335:                     r = (buffer[x+y*512]>>6)&0x1F;
336:                     b = (buffer[x+y*512]>>1)&0x1F;
337:                     RGBtoHSV( r, g, b, &i, &s, &v );
338:                     HSVtoRGB( h, s, v, &r, &g, &b );
339:                     gram[x+y*512] = (g<<11)|(r<<6)|(b<<1);
340:             }
341:     }
342:     SUPER( ssp );
343:     COMGVRAM( 0 );
344: }
345:
346: void        putS()
347: {
348:     int      r, g, b, h, s, v;
349:     unsigned short  *buffer, *gram, *another;
350:     int      x, y;
351:     int      ssp;
352:
353:     buffer = BUFFADR();
354:     gram = (unsigned short *)0xC00000;
355:     another = GETADR();
356:     ssp = SUPER( 0 );
357:     for( y=0; y<512; y++ ){
358:             for( x=0; x<512; x++ ){
359:                     g = buffer[x+y*512]>>11;
360:                     r = (buffer[x+y*512]>>6)&0x1F;
361:                     b = (buffer[x+y*512]>>1)&0x1F;
362:                     RGBtoHSV( r, g, b, &h, &s, &v );
363:                     s = (another[x+y*512]>>1)&0x1F;
364:                     HSVtoRGB( h, s, v, &r, &g, &b );
365:                     gram[x+y*512] = (g<<11)|(r<<6)|(b<<1);
366:             }
367:     }
368:     SUPER( ssp );
369:     COMGVRAM( 0 );
370: }
371:
372: void        putV()
373: {
374:     int      r, g, b, h, s, v;
375:     unsigned short  *buffer, *gram, *another;
376:     int      x, y;
377:     int      ssp;
378:
379:     buffer = BUFFADR();
380:     gram = (unsigned short *)0xC00000;
381:     another = GETADR();
382:     ssp = SUPER( 0 );
383:     for( y=0; y<512; y++ ){
384:             for( x=0; x<512; x++ ){
385:                     g = buffer[x+y*512]>>11;
386:                     r = (buffer[x+y*512]>>6)&0x1F;
387:                     b = (buffer[x+y*512]>>1)&0x1F;
388:                     RGBtoHSV( r, g, b, &h, &s, &v );
389:                     v = (another[x+y*512]>>1)&0x1F;
390:                     HSVtoRGB( h, s, v, &r, &g, &b );
391:                     gram[x+y*512] = (g<<11)|(r<<6)|(b<<1);
392:             }
393:     }
394:     SUPER( ssp );
395:     COMGVRAM( 0 );
396: }
397:
398: void        RGBtoHSV( int r, int g, int b, int *h, int *s, int *v )
399: {
400:     *v = max3( r, g, b );
401:     if( r==g && g==b ){
402:             *h = *s = 0;
403:     } else {
404:             *s = ( (*v)-min3( r, g, b ) )*31/(*v);
405:             if( *v==r ) *h = (g-b)*32/((*v)-min3( r, g, b ));
406:                     else if( *v==g ) *h = (b-r)*32/((*v)-min3( r, g, b ))+64
;
407:                             else *h = (r-g)*32/((*v)-min3( r, g, b ))+128;
408:             if( *h<0 ) *h += 192;
409:     }
410: }
411:
412: void        HSVtoRGB( int h, int s, int v, int *r, int *g, int *b )
413: {
414:     if( h>=160 ) h -= 192;
415:     if( h>=-32 && h<32 ){
416:             *r = v;
417:             if( h>=0 ){
418:                     *b = v-s*v/31;
419:                     *g = h*s*v/(31*32)+*b;
420:             } else {
421:                     *g = v-s*v/31;
422:                     *b = *g-h*s*v/(31*32);
423:             }
424:     } else if( h>=32 && h<96 ){
425:             *g = v;
426:             h -= 64;
427:             if( h>=0 ){
428:                     *r = v-s*v/31;
429:                     *b = h*s*v/(31*32)+*r;
430:             } else {
431:                     *b = v-s*v/31;
432:                     *r = *b-h*s*v/(31*32);
433:             }
434:     } else {
435:             *b = v;
436:             h -= 128;
437:             if( h>=0 ){
438:                     *g = v-s*v/31;
439:                     *r = h*s*v/(31*32)+*g;
440:             } else {
441:                     *r = v-s*v/31;
442:                     *g = *r-h*s*v/(31*32);
443:             }
444:     }
445: }
446:
447: int         max3( int r, int g, int b )
448: {
449:     int      x, y;
450:
451:     x = g+((r-g)+abs(r-g))/2;
452:     y = b+((x-b)+abs(x-b))/2;
453:     return( y );
454: }
455:
456: int         min3( int r, int g, int b )
457: {
458:     int      x;
459:
460:     x = -max3( -r, -g, -b );
461:     return( x );
462: }
```

▶まだまだ敷居の高いコンピュータミュージック。ハードディスク240Mバイト，メインメモリ12Mバイトまでようやくたどり着いた環境を無駄にしないためにも手を出してみたいですね。でも，いまはCGに染まっているので，スキャナが欲しい……。

藤沢　実(20)東京都

光線追跡による

レンズフレアのシミュレーション

Tan Akihiko 丹 明彦

ちょっと毛色の変わった画像フィルタです。LightWave3Dなどで多用されているレンズレア効果をシミュレートしてみましょう。特にアニメーション映像に適用すると効果的です。

最近，でもないが，コンピュータグラフィックの技量とは“フェイク”をいかに上手に使うかということにあるのではないかと感じている。CGは数学や物理に出てくるような数式をたくさん用いる割には，正確さを求められない。それらしく見えれば勝ち手段は問わない。それがCGのリアリティなのである。

レンズフレア(Lens Flare)とは

たとえば漫画や映画などで，日差しの強い日などに，太陽からある方向に向けて光る玉（ときに6角形）が散っているような表現をご覧になったことはないだろうか。

あるいは夜中のシーンで，自動車のヘッドライトがカメラの視野に入っていないにもかかわらず，大小のライトの虚像が一直線上に不規則に並んで映っているような現象に覚えがあることと思う。

これらをレンズフレアという。光源を表現するためのテクニックのひとつである。

かのビデオトースターの3DグラフィックソフトであるLightWave3DやMacintoshなどの画像加工ソフトであるAdobe Photoshopにも備えられている(ちなみにレンズフレアという用語はPhotoshopの同名のエフェクトから知った。登録商標みたいなもんだったらどうしよう?)。比較的簡単な操作で光源の雰囲気を出すことができるので重宝されている。

レンズフレアの起こるわけ

テクニックのひとつといってもまったく根拠がないわけではない。光源からの光がレンズを通してフィルムに届くまでに複雑な反射や屈折を繰り返した結果，フィルムのあちらこちらに光源の虚像が出現するという現象を，積極的に利用しているのである。

カメラのレンズは，収差(レンズの特性から発生する像の歪みや色のずれなど)を防ぐために大小さまざまの凸レンズや凹レンズを組み合わせて作られる。外界から入ってきた光は複雑に屈折しながらできるだけ忠実な像をフィルム上に結ぶのである。

一方，レンズはいってみれば高精度に加工されたガラス玉であるから，基本的な特性はガラスのそれである。当然，屈折だけでなく，反射もする。屈折のみを繰り返してフィルムに届いた光が本来の被写体の像になるのに対し，途中でレンズの内面や表面で反射してフィルムに届いた光は本来の被写体とは違う場所に像を結び，結果として迷惑な虚像を発生させる。

むろん，通常の被写体程度の光量であれば，光は反射と屈折を繰り返すうちに減衰してしまって虚像を作らない。しかし光源ともなると話は別で，減衰しきれず，はっきりと目に見えるほどの像が出現するのである。

この理由からカメラのレンズ設計者はレンズ内部の余分な反射を嫌い，この現象がなるべく起きないようにレンズ表面に特殊処理を加えたりレンズの組み合わせを工夫したりするのである。

戦略(光線追跡)

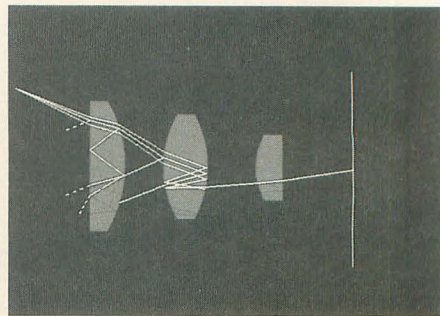
今回はこのレンズフレアを私なりの方法でシミュレートしてみる(もっとも，アルゴリズムを自分で考えただけということで，このくらいは誰かがすでにやっているはずである)。手描きではなかなか表現できない画像エフェクトのひとつとして役立つことだろう。

そこで実現のための戦略であるが，今回は3つ候補を挙げてみた。

- 経験則によるアドホックな手法
- レイトレーシング（視線追跡）
- 光線追跡(backwards ray tracing)

最初の“アドホック(ad hoc)”とは，その問題に限定された，とか，その場しのぎの，という意味の言葉で，ここでは最適な近似式を見つけ出して使うことを指している。実際のところ，データが揃っていればこの方式がもっとも高速かつリアルに表現できる。つまり現実のレンズと光源を使って，故意にレンズフレアを発生させ，光源とレンズのなす角度とレンズフレアの出現パターン(位置，大きさ，色)との関係式を編み出すことに成功すればよいのである。しかしながら私は勘が悪く，この種の問題解決方法に慣れていないこともあり，どこから手をつけていいかわからなかった。したがって残念ながら見送り。

続いてお馴染みのレイトレーシング。フィルムの各点から発生させた視線(レイ)がレンズ内部を屈折/反射したのち外に出て最初にヒットした物体の形状や属性から色を計算し，フィルムに残す。レンズの形状や構成に即して屈折や反射による光量の減衰を結果に反映させられる，きちんとした光学シミュレーションといっていいだろう(もちろんカメラメーカーで行われているのはもっともっと精密に違いないが)。ただ，レイトレーシングには間接光を処理できないという問題があり，今回のレンズフレアには少々使いにくい。

私は子供の頃に，太陽光線を鏡に反射させて部屋の壁を照らすという遊びをしたこ

光線の径路を探る

とがあるが，これはレイトレーシングでは処理できない。視線を追跡してたどり着けるのはせいぜい壁までで，そこから鏡を介して太陽まで追跡はできない。同じように，虫めがねを使って光を1点に集めるというのもレイトレーシングでは処理できない。これが間接光である。直接光源から光の当たらない部分は，正直にレイトレーシングする限りは影のままなのである。そしてレンズフレアもまさにそうした間接光の部類に入る。

この間接光を表現できるのが，今回採用することにした光線追跡なのである。レイトレーシングがフィルムから光源へ向けて視線を発生させていたのとは逆に，光源から光線を発生させてフィルムに届いたものをピックアップする。屈折や反射をシミュレートするときに用いる数学的手法はレイトレーシングと同じであるが，探索方向が異なる。

なおレイトレーシングを直訳すると光線追跡となってしまうが，レイトレーシングのアルゴリズムはむしろ視線追跡という訳語がふさわしいだろう。ここでは光線追跡といえば“逆方向の”レイトレーシングを意味することにする。

理論的には，光線追跡による間接光のシミュレーションを完璧に行おうとすれば，あらゆる光源からあらゆる方向への光の影響を計算することになるのだろう。が，もちろんコンピュータグラフィックをそんなに馬鹿正直に行う必要もないので，無駄な計算を省く研究がなされている。私も，光線追跡を限定的に使用することによってレンズフレアらしきものをでっちあげることができた。

モデル化とアルゴリズム

レンズフレアのシミュレーションでは，フィルムとレンズと光源を取り扱う。光源から発する光線のうち，第一レンズ(外界に接しているレンズ)に直接当たる光線だけを追跡してフィルムに到達するかどうかを調べればよいので，計算量はかなり削減できる。

さらに，レンズが軸対称であるという前提を利用すれば，2次元空間で光線追跡を行うことができる(図1)。これは計算量を飛躍的に減らすことができる。光源（無限遠方にある）とレンズの軸とで平面を構成すると，光線は正確にその平面上を進行するためである。

光線追跡はレイトレーシングと同様の手法によって行う(図2)。第一レンズを通過する光線を何本も放ち，レンズ内部を屈折および反射してフィルムに到達する光線を選び出す。これは総当たりといってもよく，フィルムに到達するのはほんの一部である。フィルムに到達する光線があったら，フィルムとの交点および光の強度を得ることができる。

フィルム上にレンズフレアを描画するためにはもうひとつ情報が必要である。それが光の広がり具合である。同じ光源から発した光でも，経路（つまり反射や屈折の回数や順序）によって広がり具合が変わるのである。これがレンズフレアの大小さまざまの光源像となって現れるのである。

コトを正確にしようとするのなら，コーントレーシングやビームトレーシング（光源の広がりを考慮したレイトレーシングの拡張）を行う必要があるのだが，そこはそれ，リーズナブルにいきたい私としては，

図1　レンズフレアのシミュレーション(2次元の光線追跡)

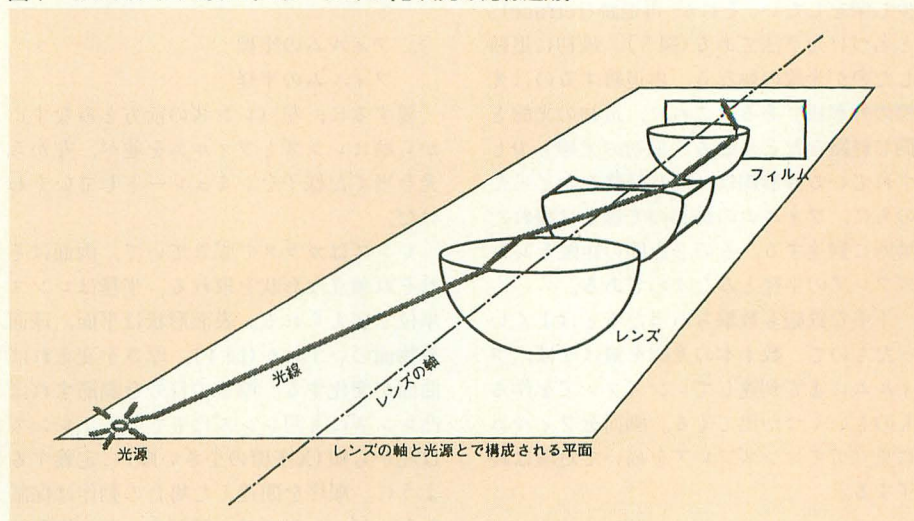



図2　光線追跡のアルゴリズム

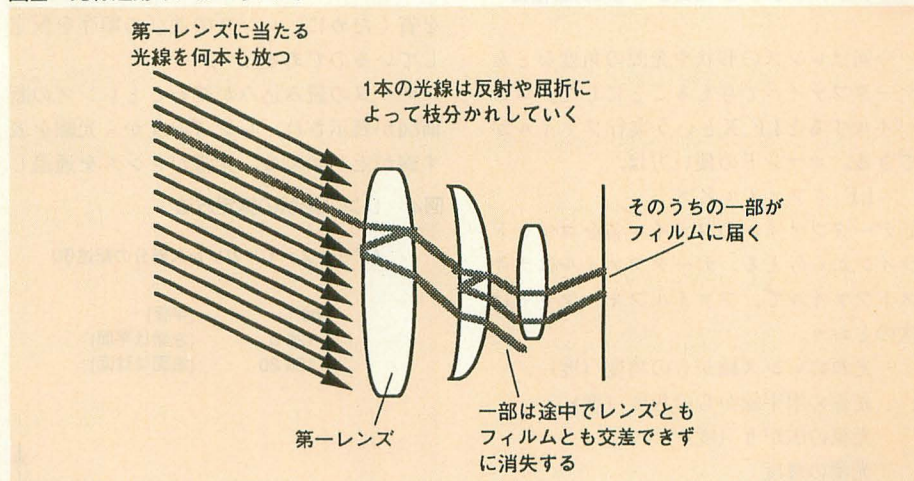
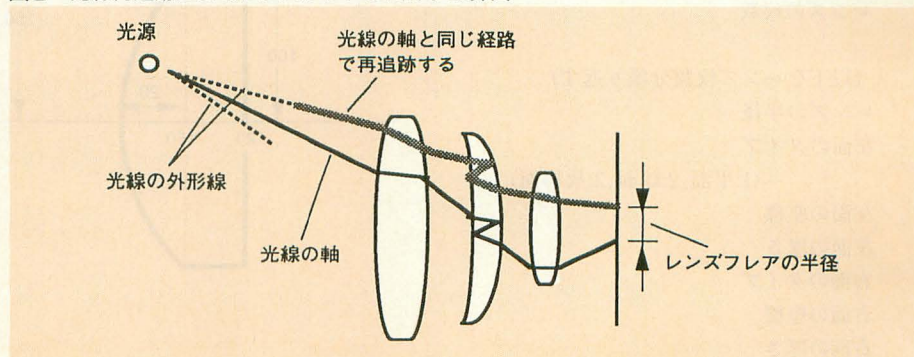



図3　光線再追跡によるレンズフレアの広がりの算出

生成されたフレアその1

その2

その3

少し楽をしたい。それが「再追跡(retrace)」と名づけた手法である(図3)。最初に追跡したのが光線の軸なら，再追跡するのは光線の外形線である。これに，最初の光線と同じ経路をたどらせる。最初の光線と少しずれている外形線は，同じ経路をたどったのちに，フィルムの最初の光線とは離れた場所に到達する。その2点間の距離をレンズフレアの半径とみなすのである。

下手な鉄砲も数撃ちゃ当たるとはよくいったもので，数十本の光線を飛ばせば，フィルムにまで到達してレンズフレアを作るものもいくつか出てくる。画面をフィルムに見立ててレンズフレアを描いて処理は終了する。

レンズフレアシミュレータの使用法

今回はレンズの形状や光源の角度などをデータファイルで与えることにした。コンパイルするとLF.Xという実行ファイルができる。コマンドの使い方は，

LF <ファイル名>

でデータファイルのファイル名をコマンドラインから与える。データファイルはテキストファイルで，ファイルフォーマットは次のとおり。

光源のレンズ軸からの角度(度)
光源の水平線からの角度(度)
光源の広がり(度)
光源の輝度

レンズの枚数

(以下をレンズ枚数分繰り返す)
レンズの半径
左面のタイプ
(1:平面,2:球面,3:放物面)
左面の座標
左面の厚さ
右面のタイプ
右面の座標
右面の厚さ
:
フィルムの座標
フィルムの半径

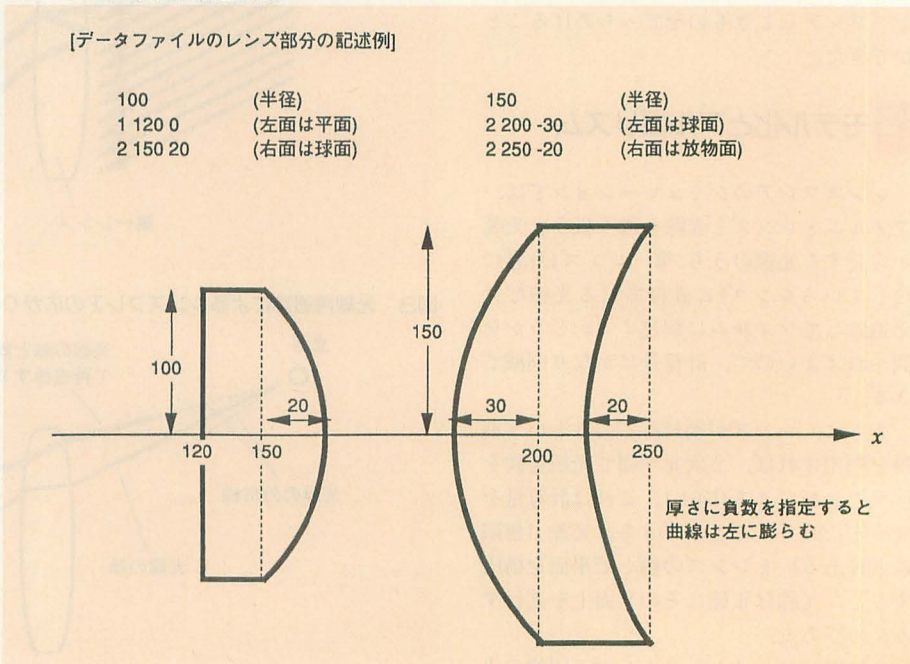
要するに，左(レンズの前方とみなす)から順にレンズとフィルムを並べ，左から光を当てた様子をシミュレートしていくわけだ。

レンズはガラスでできていて，両面はそれぞれ独立な形状を取れる。半径はレンズ単位で変えられる。表面形状は平面，球面，放物面のいずれか(図4)。厚さを変えれば曲面が変化する。厚さの符号を調節すれば凸レンズにも凹レンズにもできる。レンズは左から順(X座標の小さい順)に定義するように。順序を間違えた場合の動作は保証しないが，レンズフレアがまったく出ないだけであろう。なにしろ，不要な交点計算を省くために，レンズの並びの順序を仮定しているのである。

データの読み込みが終わるとレンズの断面図が表示され，レンズの左から光線を表す線が走り始める。光線がレンズを通過してフィルムを表す右端の線に当たればレンズフレアが発生する。光線が反射や屈折を繰り返すたびに光量は落ちていく。それは光線のグラフィック表示の色に反映される。またレンズフレアが発生した場合，光線の外形線を再追跡する。それは水色の線で表示される。

光線1本分の追跡が終了するごとにキー入力待ちになるので，スペースキーを押して次の光線に進む。レンズをひととおりスキャンし終えると，レンズフレアのレンダリングに移る。描画が終われば，キー入力待ちになるので，ESCキーを押せば終了する。

このシミュレータは単に適当にレンズを置いていくだけのものであり，実際のカメラで使用しているような精密なパラメータは設定できない。もちろん，こんなもので正しくカメラレンズとして機能するものを求めてはいけない。今回はカメラレンズの本質的な部分以外のところをシュミレートしているのである。

図4 レンズ形状の指定方法

おわりに

レンズフレアの出方は，ちょっとレンズの大きさや厚さを変えただけでも極端に変わるので，綺麗に見えるセッティングをさがすのはなかなか難しい。どうしても試行錯誤になってしまう。

実数演算を多用しているにもかかわらず，処理速度が思ったより速かったので，GUIをつけてリアルタイムにプレビューすることも不可能ではないようだ。

今回やり残したことは，

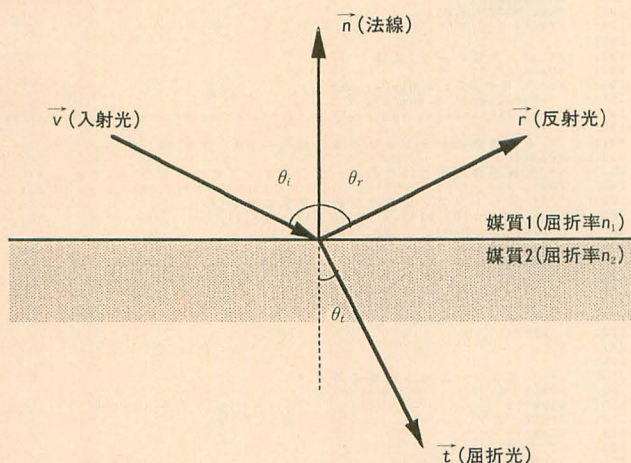
・ガラスの，光の波長ごとの屈折率や色収差の違いを利用した虹色のレンズフレア

・極端に光量が多い場合に現れる，星状のレンズフレア

・円形以外に6角形のレンズフレア（絞りの形と思われる）

などであるが，これならそれこそアドホックに法則を決めて後処理で十分対処できそうだ。

合成例

図5 （参考）反射と屈折

鏡面反射

$$\theta_i = \theta_r$$

反射ベクトル

$$\vec{r} = 2(\vec{n}\cdot\vec{v})\vec{n} + \vec{v}$$

Snell(スネル)の法則

$$\frac{\sin\theta_t}{\sin\theta_i} = \frac{n_1}{n_2}$$

屈折ベクトル

$$\vec{t} = \frac{n_1}{n_2}\vec{v} - \left(\cos\theta_t + \frac{n_1}{n_2}(\vec{n}\cdot\vec{v})\right)\vec{n}$$

保存則

反射成分 p_r, 屈折成分 p_t

$$p_r + p_t = 1$$

Fresnel(フレネル)の方程式

$$\begin{aligned} p_r &= \frac{1}{2}\left(\frac{\tan^2(\theta_i-\theta_t)}{\tan^2(\theta_i+\theta_t)} + \frac{\sin^2(\theta_i-\theta_t)}{\sin^2(\theta_i+\theta_t)}\right) \\ &= \frac{1}{2}\frac{\sin^2(\theta_i-\theta_t)}{\sin^2(\theta_i+\theta_t)}\left(1 + \frac{\cos^2(\theta_i+\theta_t)}{\cos^2(\theta_i-\theta_t)}\right) \\ &= \frac{1}{2}\frac{(g-c)^2}{(g+c)^2}\left(1 + \frac{\{c(g+c)-1\}^2}{\{c(g-c)+1\}^2}\right) \end{aligned}$$

ここで

$$\begin{aligned} c &= \cos\theta_i \\ g^2 &= \left(\frac{n_2}{n_1}\right)^2 + c^2 - 1 \end{aligned}$$

である。

ただし $g^2 < 0$ の場合は全反射となって $p_r = 1$ であり屈折成分はない。

リスト1

```
 1: /*
 2:  *  lf.c
 3:  *  - Lens Flareシミュレータ
 4:  *    レンズの処理
 5:  *    Jun. 1994 丹  明彦 (Oh!X)
 6:  */
 7:
 8: #define    __IOCS_INLINE__
 9: #include  <iocslib.h>
10: #define    __DOS_INLINE__
11: #include  <doslib.h>
12: #include  <stdio.h>
13: #include  <math.h>
14:
15: #include  "lflens.h"
16: #include  "lfrender.h"
17: #include  "iocsline.h"
18:
19: int      yaxis;
20: double  xfilm, rfilm;
21: double  phi, theta, dphi;
22: double  initpower;
23:
24: int      nlens;
25: LFLENS  lens[10];
26:
27: #define RGB(R,G,B)  (((G)<<11)|((R)<<6)|((B)<<1))
28:
29: /*
30:  *  LFadjustValue()
31:  *  - レンズデータファイルから読み込んだ
32:  *    レンズ半径と厚さの値を使って
33:  *    残りの諸定数を計算しておく。
34:  *    球面レンズでは球の半径と中心を求める。
35:  *    放物面レンズでは比例定数を求める。
36:  */
37:
38: void  LFadjustValue( LFLENS *l )
39: {
40:   LFSURFACE *ls;
41:   int        i;
42:
43:   for ( i = 0; i < 2; i++ ) {
44:     ls = &(l->s[i]);
45:     switch ( ls->type ) {
46:     case LFSURFACE_FLAT:
47:       ls->d = 0.0;
48:       ls->a = 0.0;
49:       ls->c = 0.0;
50:       break;
51:     case LFSURFACE_SPHERICAL:
52:       ls->a = ((ls->d)*(ls->d)+(l->r)*(l->r)) / (2*ls->d);
53:       ls->c = ls->x + ls->d - ls->a;
54:       break;
55:     case LFSURFACE_PARABOLIC:
56:       ls->a = -(ls->d) / ((l->r)*(l->r));
57:       ls->c = 0.0;
58:       break;
59:     }
60:   }
61:   return;
62: }
63:
64: /*
65:  *  LFreadLens()
66:  *  - レンズデータファイルを読み込む。
67:  *
68:  *  [ファイルフォーマット]
69:  *
70:  *  phi theta 光源の方向
71:  *  dphi      光源の広がり
72:  *  power     光の強さ
73:  *
74:  *  n    レンズの枚数
75:  *
76:  *  (以下をレンズ枚数分繰り返す)
77:  *  r          レンズの半径
78:  *  type x d  左面のタイプ，x座標，厚さ
79:  *  type x d  右面のタイプ，x座標，厚さ
80:  *
81:  *  x r        フィルムのx座標，半径
82:  */
```

```
 83:
 84: int LFreadLens( char *filename )
 85: {
 86:   FILE  *fp;
 87:   int   i;
 88:
 89:   if ( (fp = fopen( filename, "r" )) == NULL ) return 0;
 90:   fscanf( fp, "%lf", &phi );    phi   = phi*M_PI/180.0;
 91:   fscanf( fp, "%lf", &theta ); theta = theta*M_PI/180.0;
 92:   fscanf( fp, "%lf", &dphi );  dphi  = dphi*M_PI/180.0;
 93:   fscanf( fp, "%lf", &initpower );
 94:   fscanf( fp, "%d", &nlens );
 95:   for ( i = 0; i < nlens; i++ ) {
 96:     fscanf( fp, "%lf", &(lens[i].r) );
 97:     fscanf( fp, "%d", &(lens[i].s[0].type) );
 98:     fscanf( fp, "%lf", &(lens[i].s[0].x) );
 99:     fscanf( fp, "%lf", &(lens[i].s[0].d) );
100:     fscanf( fp, "%d", &(lens[i].s[1].type) );
101:     fscanf( fp, "%lf", &(lens[i].s[1].x) );
102:     fscanf( fp, "%lf", &(lens[i].s[1].d) );
103:     LFadjustValue( &(lens[i]) );
104:   }
105:   fscanf( fp, "%lf", &xfilm );
106:   fscanf( fp, "%lf", &rfilm );
107:   fclose( fp );
108:   return 1;
109: }
110:
111: /*
112:  *  LFdrawLens()
113:  *  - 光線追跡のグラフィック表示の背景となる
114:  *    レンズの断面図を描く。
115:  */
116:
117: void  LFdrawLens( LFLENS *l, unsigned short c )
118: {
119:   static int  ex[2][512];
120:   int         *x;
121:   LFSURFACE   *ls;
122:   int         i, j;
123:   double      y;
124:
125:   for ( i = 0; i < 2; i++ ) {
126:     ls = &(l->s[i]);
127:     x = ex[i];
128:     switch ( ls->type ) {
129:     case LFSURFACE_FLAT:
130:       for ( j = 0; j <= (int)(l->r); j++ ) {
131:         x[j] = (int)(ls->x);
132:       }
133:       break;
134:     case LFSURFACE_SPHERICAL:
135:       for ( j = 0; j <= (int)(l->r); j++ ) {
136:         y = (double)j;
137:         if ( (ls->a) > 0.0 )
138:           x[j] = (int)( sqrt( (ls->a)*(ls->a)-y*y ) + (ls->c) );
139:         else
140:           x[j] = (int)(-sqrt( (ls->a)*(ls->a)-y*y ) + (ls->c) );
141:       }
142:       break;
143:     case LFSURFACE_PARABOLIC:
144:       for ( j = 0; j <= (int)(l->r); j++ ) {
145:         y = (double)j;
146:         x[j] = (int)( (ls->x) + (ls->d) + (ls->a)*y*y );
147:       }
148:       break;
149:     }
150:   }
151:   for ( i = 0; i <= (int)(l->r); i++ ) {
152:     line( ex[0][i], yaxis-i, ex[1][i], yaxis-i, c, 0xFFFF );
153:     line( ex[0][i], yaxis+i, ex[1][i], yaxis+i, c, 0xFFFF );
154:   }
155:   return;
156: }
157:
158: /* メインルーチン */
159:
160: void  main( int argc, char *argv[] )
161: {
162:   int     i;
163:   long    sp;
164:   double  px, py;
165:   void    LFtrace( double, double, double, double, int, int, double );
166:
167:   /* レンズデータの読み込み */
168:   if ( argc != 2 ) return;
169:   if ( LFreadLens( argv[1] ) == 0 ) return;;
170:   yaxis = 256;
171:
172:   /* 画面まわりの初期設定 */
173:   CRTMOD( 8 );  /* 256色/512×512ドット */
174:   G_CLR_ON();
175:   GPALET( 0, RGB(16, 8, 0) ); /* 背景は暗いオレンジ色 */
176:   GPALET( 1, RGB( 0, 0,31) );
177:   GPALET( 2, RGB(31, 0, 0) );
178:   GPALET( 3, RGB(31, 0,31) );
179:   GPALET( 4, RGB( 0,31, 0) );
180:   GPALET( 5, RGB( 0,31,31) );
181:   GPALET( 6, RGB(31,31, 0) );
182:   GPALET( 7, RGB(31,31,31) );
183:   for ( i = 0; i < 64; i++ )
184:     GPALET( 128+i, RGB(i/2,i/2,i/2)|(i%2) );
185:   B_CUROFF();
186:   MS_INIT();
187:   MS_LIMIT(0,0,511,511);
188:   MS_CURST(256,256);
189:   MS_CUROF();
190:   SKEY_MOD(0,0,0);
191:   VPAGE( 3 );
192:
193:   /* レンズはページ1に描く */
194:   APAGE( 1 );
195:   for ( i = 0; i < nlens; i++ )
196:     LFdrawLens( &lens[i], 1 );
197:   line( (int)xfilm, yaxis-rfilm, (int)xfilm, yaxis+rfilm, 6, 0xFFFF );
198:
199:   /* 光線追跡のグラフィック表示はページ0で行う */
200:   APAGE( 0 );
201:   /* 光線追跡の密度はこのループのステップでコントロールできる */
202:   for ( i = -(int)(lens[0].r); i <= (int)(lens[0].r); i += 10 ) {
203:     /* 第一レンズと交わる光線を発生させる */
204:     px = lens[0].s[0].x - 100.0*cos(phi);
205:     py = (double)i - 100.0*sin(phi);
206:     /* 光線追跡 */
207:     LFtrace( px, py, cos(phi), sin(phi), 0, 0, initpower );
208:     /* スペースキーが押されるまで待つ */
209:     while ( (BITSNS(0x06)&32) == 0 );
210:     while ( (BITSNS(0x06)&32) );
211:     /* ページ0だけを消す */
212:     WIPE();
213:   }
214:
215:   /* シミュレーション画面を消す */
216:   APAGE( 1 );
217:   WIPE();
218:   APAGE( 0 );
219:   WIPE();
220:
221:   /* 背景を黒に戻す */
222:   GPALET( 0, RGB(0,0,0) );
223:
224:   /* 結果のレンダリング */
225:   sp = SUPER( 0 );
226:   LFrender();
227:   SUPER( sp );
228:
229:   /* ESCキーが押されるまで待つ */
230:   while ( (BITSNS(0x00)&2) == 0 );
231:   while ( (BITSNS(0x00)&2) );
232:
233:   /* 後始末 */
234:   B_CURON();
235:   CRTMOD( 16 );
236:   KFLUSHIO( 0xFF );
237:
238:   return;
239: }
```

リスト2

```
 1: /*
 2:  *  lftrace.c
 3:  *  - Lens Flareシミュレータ
 4:  *    光線追跡
 5:  *    Jun. 1994 丹  明彦 (Oh!X)
 6:  */
 7:
 8: #define   __IOCS_INLINE__
 9: #include  <iocslib.h>
10: #include  <math.h>
11:
12: #include  "lflens.h"
13: #include  "lfrender.h"
14: #include  "iocsline.h"
15:
16: /* 数値誤差許容範囲 */
17: #define TOLERANCE 1.0e-9
18:
19: /* 「ほぼ0」の判定 */
20: #define ZERO(X) (((X)-TOLERANCE)*((X)+TOLERANCE)<0.0)
21:
22: /* 単位ベクトルを得る */
23: #define NORMALIZE(X,Y)  {¥
24:   double  __norm;¥
25:   __norm = sqrt((X)*(X)+(Y)*(Y));¥
26:   (X)/=__norm;¥
27:   (Y)/=__norm;}
28:
29: /* 光線追跡打ち切りの条件 */
30: #define MAXTRACELEVEL 20  /* 光線追跡の最深レベル */
31: #define MINTRACEPOWER 0.01  /* 光線追跡の最低強度 */
32:
33: /* 屈折率(ガラス) */
34: #define REFRACTIONINDEX 1.5
35:
36: extern int     yaxis;
37: extern double xfilm, rfilm;
38: extern double dphi;
39:
40: extern int     nlens;
41: extern LFLENS lens[10];
42:
43: int      tracehistory[ MAXTRACELEVEL ];
44: int      tracepath[ MAXTRACELEVEL ];
45: double  px0, py0, vx0, vy0;
46:
47: /*
48:  *  LFtrace()
49:  *  - 光線追跡
50:  *    光源からフィルムに到達するまで光線を追跡する
51:  *    関数内で1本の光線は反射成分と屈折成分で
52:  *    最大2本に分岐するため、再帰呼び出しにより
53:  *    可能なすべての経路を探索する。
54:  *    経路を逐一記録し(どのレンズ面に対して反射/屈折した)、
55:  *    光線がフィルム面に到達したときに
56:  *    LFretrace()によって再追跡するための情報となる。
57:  */
58:
59: void  LFtrace( double px, double py,  /* 始点(位置ベクトル) */
60:       double vx, double vy,  /* 光線(単位ベクトル) */
61:       int space,             /* 探索空間 */
62:       int tracelevel,        /* 光線追跡のレベル，深くなった時に打ち切る
*/
63:       double tracepower )    /* 光線追跡の強度，弱くなった時にも打ち切る
*/
64: {
65:   LFLENS    *ll;
66:   LFSURFACE *ls;
67:   double    t = 10000000000.0, tmpt;
68:   double    tmpx, tmpy;
69:   double    sx, sy; /* 交点の位置ベクトル */
70:   double    nx, ny; /* 法線ベクトル(単位ベクトル) */
71:   double    rx, ry; /* 反射方向ベクトル(単位ベクトル) */
72:   double    tx, ty; /* 屈折方向ベクトル(単位ベクトル) */
73:   int       i, hit = -1;
74:   double    A, B, C, D;
75:   double    nv;
76:   int       rflag, tflag;
77:   double    rpow, tpow;
78:   double    g, ci, ct;
```

▶「CGA入門キット“GENIE”」は熱い！　熱すぎる！　こ，こいつは「ガンプラ」世代には燃える！　いかす，いかすぞDōGA。連載1回休みも許す。さあ，今日も帰って“GENIE”だ。……あ，しまった，弟が先にハマッていた。　桐生　保(25)三重県

```
 79:    double     nl, nt;
 80:    int        I;
 81:    int        LFretrace( double*, double );
 82:
 83:    /* トレースの階層が深くなると打ち切る */
 84:    if ( tracelevel >= MAXTRACELEVEL ) return;
 85:
 86:    if ( tracelevel == 0 ) {
 87:      px0 = px;
 88:      py0 = py;
 89:      vx0 = vx;
 90:      vy0 = vy;
 91:    }
 92:
 93:    /* トレースの強度が弱くなると打ち切る */
 94:    if ( tracepower < MINTRACEPOWER ) return;
 95:
 96:    /* 探索空間に面するレンズ面との交点計算 */
 97:    for ( i = space-1; i <= space; i++ ) {
 98:      if ( i == -1 ) continue;  /* 左端 */
 99:      if ( i == (2*nlens) ) {    /* 右端(フィルムとの交点) */
100:        /* 交点なし */
101:        if ( ZERO( vx ) ) continue;
102:        /* 交点までの距離 */
103:        tmpt = (xfilm - px)/vx;
104:        /* 交点は始点の後ろ */
105:        if ( tmpt < TOLERANCE ) continue;
106:        /* 交点はフィルム内か? */
107:        tmpy = py + tmpt*vy;
108:        if ( (tmpy-rfilm)*(tmpy+rfilm) > TOLERANCE ) continue;
109:        /* 一番近いか? */
110:        if ( tmpt > t ) continue;
111:        /* 交点候補の更新 */
112:        t = tmpt;
113:        hit = i;
114:        /* 交点の算出 */
115:        sx = xfilm;
116:        sy = tmpy;
117:        /* 法線の算出 */
118:        nx = -1.0;
119:        ny = 0.0;
120:        continue;
121:      }
122:      ll = &(lens[i/2]);
123:      ls = &(ll->s[i%2]);
124:      switch ( ls->type ) {
125:      case LFSURFACE_FLAT:
126:        /* 交点なし */
127:        if ( ZERO( vx ) ) continue;
128:        /* 交点までの距離 */
129:        tmpt = (ls->x - px)/vx;
130:        /* 交点は始点の後ろ */
131:        if ( tmpt < TOLERANCE ) continue;
132:        /* 交点はレンズ内か? */
133:        tmpy = py + tmpt*vy;
134:        if ( (tmpy - (ll->r))*(tmpy + (ll->r)) > TOLERANCE ) continue;
135:        /* 一番近いか? */
136:        if ( tmpt > t ) continue;
137:        /* 交点候補の更新 */
138:        t = tmpt;
139:        hit = i;
140:        /* 交点の算出 */
141:        sx = ls->x;
142:        sy = tmpy;
143:        /* 法線の算出 */
144:        if ( i%2 == 0 ) {
145:          nx = -1.0;
146:          ny = 0.0;
147:        } else {
148:          nx = 1.0;
149:          ny = 0.0;
150:        }
151:        break;
152:      case LFSURFACE_SPHERICAL:
153:        B = px*vx-(ls->c)*vx+py*vy;
154:        C = ((ls->c)-px)*((ls->c)-px)+py*py-(ls->a)*(ls->a);
155:        /* 判別式 */
156:        D = B*B - C;
157:        if ( D < TOLERANCE ) continue;  /* 交点なし */
158:        /* 解を求める */
159:        D = sqrt( D );
160:        tmpt = -B-D;
161:        if ( tmpt < TOLERANCE ) { /* 手前の交点は始点の後ろ */
162:          tmpt = -B+D;
163:          if ( tmpt < TOLERANCE ) continue; /* 奥の交点も始点の後ろ */
164:        }
165:        /* 交点はレンズ内か? */
166:        tmpx = px + tmpt*vx;
167:        if ( (tmpx - (ls->x))*(tmpx - ((ls->x)+(ls->d))) > TOLERANCE ) con
tinue;
168:        /* 一番近いか? */
169:        if ( tmpt > t ) continue;
170:        /* 更新 */
171:        t = tmpt;
172:        hit = i;
173:        /* 交点の算出 */
174:        sx = tmpx;
175:        sy = py + t*vy;
176:        /* 法線の算出 */
177:        nx = sx-(ls->c);
178:        ny = sy;
179:        NORMALIZE(nx,ny);
180:        break;
181:      case LFSURFACE_PARABOLIC:
182:        A = (ls->a)*vy*vy;
183:        B = 2.0*(ls->a)*py*vy-vx;
184:        C = (ls->a)*py*py-px+(ls->x)+(ls->d);
185:        /* 光線が軸に平行な場合(vy==0)は2次方程式でなくなる */
186:        if ( ZERO(A) ) {
187:          if ( ZERO(B) ) continue;  /* 交点なし */
188:          tmpt = -C/B;
189:        } else {
190:          /* 判別式 */
191:          D = B*B - 4*A*C;
192:          if ( D < TOLERANCE ) continue;  /* 交点なし */
193:          /* 解を求める */
194:          D = sqrt( D );
195:          tmpt = (A>0.0)?((-B-D)/(2.0*A)):((-B+D)/(2.0*A));
196:          if ( tmpt < TOLERANCE ) { /* 手前の交点は始点の後ろ */
197:            tmpt = (A>0.0)?((-B+D)/(2.0*A)):((-B-D)/(2.0*A));
198:            if ( tmpt < TOLERANCE ) continue; /* 奥の交点も始点の後ろ */
199:          }
200:        }
201:        /* 交点はレンズ内か? */
202:        tmpy = py + tmpt*vy;
203:        if ( (tmpy - (ll->r))*(tmpy + (ll->r)) > TOLERANCE ) continue;
204:        /* 一番近いか? */
205:        if ( tmpt > t ) continue;
206:        /* 更新 */
207:        t = tmpt;
208:        hit = i;
209:        /* 交点の算出 */
210:        sx = px + t*vx;
211:        sy = tmpy;
212:        /* 法線の算出 */
213:        nx = 1.0;
214:        ny = -2.0*(ls->a)*sy;
215:        NORMALIZE(nx,ny);
216:        break;
217:      }
218:      continue;
219:    }
220:
221:    /* 光線の明るさ */
222:    I = (int)(64.0*tracepower);
223:    if ( I > 63) I = 63;
224:    I += 128;
225:
226:    /* 交点が存在しなかった場合 */
227:    if ( hit == -1 ) {
228:      sx = px + vx*32.0;
229:      sy = py + vy*32.0;
230:      line( (int)px, (int)py+yaxis, (int)sx, (int)sy+yaxis, I, 0xF0F0 );
231:      return;
232:    }
233:
234:    /* 光線を描く */
235:    line( (int)px, (int)py+yaxis, (int)sx, (int)sy+yaxis, I, 0xFFFF );
236:
237:    tracepath[tracelevel] = hit;
238:
239:    /* フィルムに到達 */
240:    if ( hit == 2*nlens ) {
241:      tracehistory[tracelevel] = 3;
242:      flareposition[nflare] = (int)sy;
243:      flarepower[nflare] = I-128;
244:      i = LFretrace( &sy, dphi );
245:      if ( i == -1 ) {
246:        i = LFretrace( `&sy, -dphi );
247:        if ( i == -1 ) {
248:          sy = (double)flareposition[nflare];
249:        }
250:      }
251:      flareradius[nflare] = (int)sy - flareposition[nflare];
252:      if ( flareradius[nflare] < 0 ) flareradius[nflare] *= -1;
253:      nflare++;
254:      return;
255:    }
256:
257:    /* 法線をレンズ外側に向ける */
258:    if ( (((hit%2) == 0) && (nx > 0.0)) || (((hit%2) == 1) && (nx < 0.0))
) {
259:      nx = -nx;
260:      ny = -ny;
261:    }
262:
263:    /* 反射方向のベクトルを求める(r = v - 2(n.v)n) */
264:    nv = nx*vx + ny*vy;
265:    rx = vx - 2.0*nv*nx;
266:    ry = vy - 2.0*nv*ny;
267:
268:    /* 反射と屈折 */
269:    if ( nv < 0.0 ) { /* 外(空気)から内(レンズ)へ */
270:      nl = REFRACTIONINDEX;
271:      ci = -nv;
272:      g = nl*nl + ci*ci - 1.0;
273:      if ( g < 0.0 ) {  /* 屈折成分なし */
274:        rflag = 1;
275:        rpow = 1.0;
276:        tflag = 0;
277:        tpow = 0.0;
278:      } else {
279:        g = sqrt( g );
280:        /* フレネルの法則にしたがって反射光の強度を求める */
281:        rflag = 1;
282:        rpow = 0.5 * ((g-ci)*(g-ci))/((g+ci)*(g+ci)) * (1.0 + ((ci*(g+ci)-
1.0)/(ci*(g-ci)+1.0)));
283:        /* 保存則にしたがって屈折光の強度を求める */
284:        tflag = 1;
285:        tpow = 1.0 - rpow;
286:      }
287:    } else {  /* 内から外へ */
288:      nl = 1.0/REFRACTIONINDEX;
289:      ci = nv;
290:      g = nl*nl + ci*ci - 1.0;
291:      if ( g < 0.0 ) {  /* 屈折成分なし */
292:        rflag = 1;
293:        rpow = 1.0;
294:        tflag = 0;
295:        tpow = 0.0;
296:      } else {
297:        g = sqrt( g );
298:        rflag = 1;
299:        rpow = 0.5 * ((g-ci)*(g-ci))/((g+ci)*(g+ci)) * (1.0 + ((ci*(g+ci)-
1.0)/(ci*(g-ci)+1.0)));
300:        tflag = 1;
301:        tpow = 1.0 - rpow;
302:      }
303:    }
304:
305:    if ( rflag ) {  /* 反射光の追跡 */
306:      tracehistory[tracelevel] = 1;
307:      LFtrace( sx, sy, rx, ry, space, tracelevel+1, tracepower*rpow );
308:    }
309:
310:    if ( tflag ) {  /* 屈折光の追跡 */
311:      nt = REFRACTIONINDEX;
312:      if ( nv < 0.0 ) { /* 外から内へ */
313:        /* 屈折ベクトルの算出 */
314:        ct = sqrt( nt*nt-1.0+nv*nv )/nt;
315:        tx = vx/nt + (-(ct+nv/nt))*nx;
316:        ty = vy/nt + (-(ct+nv/nt))*ny;
317:        tracehistory[tracelevel] = 2;
318:        LFtrace( sx, sy, tx, ty, 2*hit-space+1, tracelevel+1, tracepower*t
pow );
319:      } else {    /* 内から外へ */
320:        ct = sqrt( 1.0-nt*nt*(1.0-nv*nv) );
321:        tx = vx*nt + (ct-nv*nt)*nx;
322:        ty = vy*nt + (ct-nv*nt)*ny;
323:        tracehistory[tracelevel] = 2;
324:        LFtrace( sx, sy, tx, ty, 2*hit-space+1, tracelevel+1, tracepower*t
pow );
325:      }
326:    }
327:
328:    return;
329: }
330:
331: /*
332:  *  LFretrace()
333:  *  - 光線の再追跡による光源像の大きさの調査
334:  *    LFtrace()で光線がフィルムに到達しるごとに呼び出される。
335:  *    光線とdphiだけずれた角度で出発し、記録されている経路と
336:  *    同じ経路をたどってフィルム上の位置をresultに返す。
```

▶ 7月号の付録ディスクは面白いじゃないですか。2年前のDōGAのディスクは難しくてなんだかやりにくかったけど，今回のはとてもやさしい感じがします。

宮本　久嗣(19)広島県

```
337:  *    経路は1本なので再帰呼び出しは使わない。
338:  *    途中で再追跡に失敗した場合は戻り値として-1を返す。
339:  */
340:
341: int LFretrace( result, dphi )
342: double  *result, dphi;
343: {
344:   int        i, hit;
345:   LFLENS     *ll;
346:   LFSURFACE *ls;
347:   double     t;
348:   double     px, py;
349:   double     vx, vy;
350:   double     nx, ny;
351:   double     sx, sy;
352:   double     rx, ry;
353:   double     tx, ty;
354:   double     A, B, C, D;
355:   double     nv;
356:   double     g, ci, ct;
357:   double     nl, nt;
358:
359:   /* 保存しておいた始点 */
360:   px = px0;
361:   py = py0;
362:   /* 光線方向を回転させる */
363:   vx =  cos(dphi)*vx0 + sin(dphi)*vy0;
364:   vy = -sin(dphi)*vx0 + cos(dphi)*vy0;
365:
366:   for ( i = 0; i < MAXTRACELEVEL; i++ ) {
367:     hit = tracepath[i];
368:     if ( tracehistory[i] == 3 ) { /* フィルムとの交点 */
369:       /* 交点なし */
370:       if ( ZERO( vx ) ) return -1;
371:       /* 交点までの距離 */
372:       t = (xfilm - px)/vx;
373:       /* 交点は始点の後ろ */
374:       if ( t < TOLERANCE ) return -1;
375:       /* 交点はフィルム内か? */
376:       sy = py + t*vy;
377:       if ( (sy-rfilm)*(sy+rfilm) > TOLERANCE ) return -1;
378:       /* 光線を描く */
379:       line( (int)px, (int)py+yaxis, (int)xfilm, (int)sy+yaxis, 5, 0xFFFF
);
380:       /* 交点を返す */
381:       *result = sy;
382:       return 0;
383:     }
384:     ll = &(lens[hit/2]);
385:     ls = &(ll->s[hit%2]);
386:     switch ( ls->type ) {
387:     case LFSURFACE_FLAT:
388:       /* 交点なし */
389:       if ( ZERO( vx ) ) return -1;
390:       /* 交点までの距離 */
391:       t = (ls->x - px)/vx;
392:       /* 交点は始点の後ろ */
393:       if ( t < TOLERANCE ) return -1;
394:       /* 交点はレンズ内か? */
395:       sy = py + t*vy;
396:       if ( (sy - (ll->r))*(sy + (ll->r)) > TOLERANCE ) return -1;
397:       /* 交点の算出 */
398:       sx = ls->x;
399:       /* 法線の算出 */
400:       if ( hit%2 == 0 ) {
401:         nx = -1.0;
402:         ny = 0.0;
403:       } else {
404:         nx = 1.0;
405:         ny = 0.0;
406:       }
407:       break;
408:     case LFSURFACE_SPHERICAL:
409:       B = px*vx-(ls->c)*vx+py*vy;
410:       C = ((ls->c)-px)*((ls->c)-px)+py*py-(ls->a)*(ls->a);
411:       /* 判別式 */
412:       D = B*B - C;
413:       if ( D < TOLERANCE ) return -1; /* 交点なし */
414:       /* 解を求める */
415:       D = sqrt( D );
416:       t = -B-D;
417:       if ( t < TOLERANCE ) {  /* 手前の交点は始点の後ろ */
418:         t = -B+D;
419:         if ( t < TOLERANCE ) return -1; /* 奥の交点も始点の後ろ */
420:       }
421:       /* 交点はレンズ内か? */
422:       sx = px + t*vx;
423:       if ( (sx - (ls->x))*(sx - ((ls->x)+(ls->d))) > TOLERANCE ) return
-1;
424:       /* 交点の算出 */
425:       sy = py + t*vy;
426:       /* 法線の算出 */
427:       nx = sx-(ls->c);
428:       ny = sy;
429:       NORMALIZE(nx,ny);
430:       break;
431:     case LFSURFACE_PARABOLIC:
432:       A = (ls->a)*vy*vy;
433:       B = 2.0*(ls->a)*py*vy-vx;
434:       C = (ls->a)*py*py-px+(ls->x)+(ls->d);
435:       /* 光線が軸に平行な場合(vy==0)は2次方程式でなくなる */
436:       if ( ZERO(A) ) {
437:         if ( ZERO(B) ) return -1; /* 交点なし */
438:         t = -C/B;
439:       } else {
440:         /* 判別式 */
441:         D = B*B - 4*A*C;
442:         if ( D < TOLERANCE ) return -1; /* 交点なし */
443:         /* 解を求める */
444:         D = sqrt( D );
445:         t = (A>0.0)?((-B-D)/(2.0*A)):((-B+D)/(2.0*A));
446:         if ( t < TOLERANCE ) {  /* 手前の交点は始点の後ろ */
447:           t = (A>0.0)?((-B+D)/(2.0*A)):((-B-D)/(2.0*A));
448:           if ( t < TOLERANCE ) return -1; /* 奥の交点も始点の後ろ */
449:         }
450:       }
451:       /* 交点はレンズ内か? */
452:       sy = py + t*vy;
453:       if ( (sy - (ll->r))*(sy + (ll->r)) > TOLERANCE ) return -1;
454:       /* 交点の算出 */
455:       sx = px + t*vx;
456:       /* 法線の算出 */
457:       nx = 1.0;
458:       ny = -2.0*(ls->a)*sy;
459:       NORMALIZE(nx,ny);
460:       break;
461:     }
462:     /* 光線を描く */
463:     line( (int)px, (int)py+yaxis, (int)sx, (int)sy+yaxis, 5, 0xFFFF );
464:
465:     /* 法線をレンズ外側に向ける */
466:     if ( (((hit%2) == 0) && (nx > 0.0)) || (((hit%2) == 1) && (nx < 0.0)
) ) {
467:       nx = -nx;
468:       ny = -ny;
469:     }
470:
471:     /* 反射方向のベクトルを求める */
472:     nv = nx*vx + ny*vy;
473:     rx = vx - 2.0*nv*nx;
474:     ry = vy - 2.0*nv*ny;
475:
476:     if ( tracehistory[i] == 1 ) { /* 反射光の追跡 */
477:       px = sx;
478:       py = sy;
479:       vx = rx;
480:       vy = ry;
481:       continue; /* 次の経路探索へ */
482:     } else {  /* 屈折光の追跡 */
483:       if ( nv < 0.0 ) { /* 外から内へ */
484:         nl = REFRACTIONINDEX;
485:         ci = -nv;
486:       } else {  /* 内から外へ */
487:         nl = 1.0/REFRACTIONINDEX;
488:         ci = nv;
489:       }
490:       g = nl*nl + ci*ci - 1.0;
491:       if ( g < 0.0 ) return -1; /* 屈折成分なし */
492:
493:       nt = REFRACTIONINDEX;
494:       if ( nv < 0.0 ) { /* 外から内へ */
495:         ct = sqrt( nt*nt-1.0+nv*nv )/nt;
496:         tx = vx/nt + (-(ct+nv/nt))*nx;
497:         ty = vy/nt + (-(ct+nv/nt))*ny;
498:       } else {    /* 内から外へ */
499:         ct = sqrt( 1.0-nt*nt*(1.0-nv*nv) );
500:         tx = vx*nt + (ct-nv*nt)*nx;
501:         ty = vy*nt + (ct-nv*nt)*ny;
502:       }
503:       px = sx;
504:       py = sy;
505:       vx = tx;
506:       vy = ty;
507:       continue; /* 次の経路探索へ */
508:     }
509:   }
510:
511:   return -1;
512: }
```

リスト3

```
 1: /*
 2:  *   lfrender.c
 3:  *   - Lens Flareシミュレータ
 4:  *     レンダリング
 5:  *     Jun. 1994 丹  明彦 (Oh!X)
 6:  */
 7:
 8: #include  <math.h>
 9:
10: /* 水平線に対する光源方向の角度 */
11: extern double theta;
12:
13: /* G-RAMを2次元配列でアクセスするための宣言
14:     (括弧の位置がポイント) */
15: unsigned short  (*gram)[512] = 0xC00000;
16:
17: /* レンズフレアの情報
18:  * LFtrace()とLFretrace()によって蓄積される */
19: int nflare = 0;
20: int flareposition[4000];
21: int flarepower[4000];
22: int flareradius[4000];
23:
24: /*
25:  *   LFpaintCircle()
26:  *   - レンズフレアの描画
27:  *     所定の位置、明るさ、半径で円を塗りつぶす。
28:  *     値は重ね合わせられる。
29:  *     256色モードでパレットコード128～191が
30:  *     グレイスケールに割り当ててあることが前提。
31:  */
32:
33: void  LFpaintCircle( x, y, r, p )
34: int x, y, r, p;
35: {
36:   int i, j, t, c;
37:
38:   for ( i = -r; i <= r; i++ ) {
39:     t = sqrt(r*r-i*i);
40:     for ( j = -t; j <= t; j++ ) {
41:       c = gram[y+i][x+j];
42:       if ( c == 0 )
43:         c = p;
44:       else
45:         c = c - 128 + p;
46:       if ( c > 63 ) c = 63;
47:       gram[y+i][x+j] = c + 128;
48:     }
49:   }
50:   return;
```

▶「ニュートン」の日本語版が「ガリレオ」ですか……。どんどん古くなっていくような。新しいのが出たら「アリストテレス」とでもなるんでしょうか。

岡本　訓(18)神奈川県

```
51: }
52:
53: /*
54:  *  LFrender()
55:  *  - レンズフレアのレンダリング
56:  *     光線追跡によって得たレンズフレア情報に従って
57:  *     レンダリングを行う。
58:  *     G-RAMを直接アクセスするため
59:  *     スーパーバイザモードで呼び出すこと。
60:  */
61:
```

```
62: void  LFrender()
63: {
64:   int i, x, y;
65:
66:   for ( i = 0; i < nflare; i++ ) {
67:     x = 256+(int)((double)flareposition[i]*cos(theta));
68:     y = 256-(int)((double)flareposition[i]*sin(theta));
69:     LFpaintCircle( x, y, flareradius[i], flarepower[i] );
70:   }
71:   return;
72: }
```

リスト4

```
 1: /*
 2:  *  lflens.h
 3:  *  - Lens Flareシミュレータ
 4:  *    レンズの定義
 5:  *    Jun. 1994 丹  明彦 (Oh!X)
 6:  */
 7:
 8: #ifndef __LFLENS_H__
 9: #define __LFLENS_H__
10:
11: /* レンズの面 */
12: typedef struct  _lfsurface {
13:   int      type; /* 面形状の種類 */
14:   double  x;     /* 中心のx座標(y座標は常に0) */
15:   double  d;     /* 厚さ */
16:   double  a;     /* (球面)球の半径,(放物面)比例定数 */
17:   double  c;     /* (球面)球の中心のx座標 */
18: } LFSURFACE;
19:
20: /* 面形状 */
21: #define LFSURFACE_FLAT       1 /* 平面 */
22: #define LFSURFACE_SPHERICAL 2 /* 球面 */
23: #define LFSURFACE_PARABOLIC 3 /* 放物面 */
24:
25: /* レンズ */
26: typedef struct  _lflens {
27:   double     r;     /* 半径 */
28:   LFSURFACE s[2]; /* s[0]は左面,s[1]は右面 */
29: } LFLENS;
30:
31: #endif  /* __LFLENS_H__ */
```

リスト5

```
 1: /*
 2:  *  lfrender.h
 3:  *  - Lens Flareシミュレータ
 4:  *    レンダリング
 5:  *    Jun. 1994 丹  明彦 (Oh!X)
 6:  */
 7:
 8: #ifndef __LFRENDER_H__
 9: #define __LFRENDER_H__
10:
11: extern int  nflare;
12: extern int  flareposition[4000];
13: extern int  flarepower[4000];
14: extern int  flareradius[4000];
15:
16: void  LFrender();
17:
18: #endif  /* __LFRENDER_H__ */
```

リスト6

```
 1: /*
 2:  *  iocsline.h
 3:  *  - X-BASICライクな簡易描画関数
 4:  *    Jun. 1994 A.Tan (Oh!X)
 5:  */
 6:
 7: #ifndef __IOCSLINE_H__
 8: #define __IOCSLINE_H__
 9:
10: #define line(X1,Y1,X2,Y2,COLOR,LINESTYLE) {¥
11:   struct _lineptr _l;¥
12:   _l.x1 = X1;¥
13:   _l.y1 = Y1;¥
14:   _l.x2 = X2;¥
15:   _l.y2 = Y2;¥
16:   _l.color = COLOR;¥
17:   _l.linestyle = LINESTYLE;¥
18:   LINE( &_l );}
19:
20: #endif  /* __IOCSLINE_H__ */
```

リスト7

```
 1: # Makefile for lf, a lens flare simulator
 2: # Jun. 1994      A.Tan (Oh!X)
 3:
 4: CCOPTS          = -O -Wall
 5: #CCOPTS         = -m68040 -O -Wall
 6:
 7: %.o: %.s
 8:         has -u -w $<
 9:
10: %.o: %.c
11:         gcc $(CCOPTS) -c $<
12:
13: all: lf.x
14:
15: lf.x: lf.o lftrace.o lfrender.o
16:         gcc lf.o lftrace.o lfrender.o
17:
18: lf.o: lf.c lflens.h lfrender.h iocsline.h
19: lftrace.o: lftrace.c lflens.h lfrender.h iocsline.h
20: lfrender.o: lfrender.c
21:
22: clean:
23:         rm -f lf.x lf.o lftrace.o lfrender.o
```

リスト8

```
 1: 20 30 10 50
 2:
 3: 3
 4:
 5: 100
 6: 1 100 0
 7: 2 120 25
 8:
 9: 40
10: 2 200 -5
11: 3 220 5
12:
13: 60
14: 3 300 -10
15: 1 320 0
16:
17: 400 150
```

リスト9

```
 1: 20 45 3 50
 2:
 3: 7
 4:
 5: 100
 6: 3 100 -8
 7: 3 105 8
 8:
 9: 95
10: 3 110 8
11: 1 125 0
12:
13: 90
14: 3 138 -12
15: 1 140 0
16:
17: 85
18: 3 200 -20
19: 3 210 -15
20:
21: 60
22: 3 300 -5
23: 1 305 0
24:
25: 60
26: 1 306 0
27: 3 315 -5
28:
29: 70
30: 3 400 -10
31: 3 405 -8
32:
33: 500 150
```

リスト10

```
 1: 25 30 10 50
 2:
 3: 3
 4:
 5: 100
 6: 1 100 0
 7: 2 120 25
 8:
 9: 40
10: 2 200 -5
11: 3 220 5
12:
13: 60
14: 3 300 -10
15: 1 320 0
16:
17: 400 150
```

▶「コットン」の湯飲みが欲しい方へ。メガドライブ版を買うともれなく貰えます。ただし,締め切りがあるので注意してください。　平　勝久(20)大阪府

簡易ドローの拡張

PICTを使う

Ishigami Tatsuya 石上 達也

SX-WINDOWのグラフィックデータ構造なのになかなか解説されることのないPICT。ここではPICTデータを簡単に作成するために簡易ドローを拡張して簡単なエディタを作成してみましょう。

昔「開発キット」のリソースエディタに対してひどい誤解をしてしまったことがありました。リソース内容の表示はできるのに，なぜか編集ができない，みたいなことを書いてしまいましたが，結局，私の誤解でリソースエディタはちゃんとリソースを作れるだけの機能を持っていました。

よく使われるPAT4やDLTLは専用のエディット機能を備えており，グラフィカルな編集作業が可能となっています。メニュー文字列もテンプレートが用意されているので，空欄に必要事項を埋め込んでいくだけで作成することができます。

しかし，PICTを表示することはできても，なぜか編集することはできませんでした。

PICTとは

SX-WINDOWがver.3.1になって広告にGScriptという単語が使用されるようになりました。おそらくGraphic Scriptの略だと思われますが，これはいわゆるドロー系のグラフィックデータ構造のことを表します。GScriptといっても，どういうわけかリソースタイプの名前はPICTです。以下PICTで統一します。これはSX-WINDOW上でのドローデータの基本形式として扱われていくことでしょう。

Easydrawからクリップボードにデータを転送すると，PICTとDRAWとの2通りのデータ形式で送られます。おそらく，DRAWはEasydraw特有のデータ形式でシャーペンなどに図形を転送するため，これとあわせてPICT形式のデータも転送するのでしょう。

私はEasydrawを持っていないのでわからないのですが，本誌1993年11月号を見ると，複雑な図形をシャーペンに張り込んだ場合，若干絵の品質が損なわれているのがわかります（77ページのEasydrawで描いた憂鬱くんと「そのままペースト」でシャーペンへ転送された憂鬱くんを比べてください）。おそらく，PICTよりもDRAWのほうが扱えるデータの種類が豊富なのでしょう。ですから，

1)　Easydrawから出てEasydrawへ戻すデータ

独自形式で転送するので品質も落とさずにやりとりできる。

2)　それ以外のアプリケーションに転送するデータ

SX-WINDOW共通の仕様であるPICT形式で，なるべく同じような図形になるようにしてデータを渡す。多少，図形が違っても目をつぶる。

と，なるのでしょう（編注：SX-WINDOW ver.3.1ではシャーペンもDRAW型に対応しています）。

Macintoshではたいていのドローツールにデータの保存形式を選択する機能がついています。 Macintoshのスクリプトデータは正直にPICTというのですが，多くのドローツールでこの形式を選択すると，

「絵の一部が損なわれてしまいます。よろしいですか？
　はい　　いいえ　　キャンセル」

のようなダイアログを開いて聞いてきます。

このように，Macintoshでは多少絵の品質を落としてでも，ほかのアプリケーションとのデータ互換性を優先させたい場合が多々あります。SX-WINDOWでも将来（?），さまざまなドローツールが発売された場合，PICTがそれらの間で標準的なフォーマットとして用いられるものと思います。

2つのドロー形式

Easydraw.xで作成した書類をメニューでコピーしてクリップボードに送り，リスト表示してみると，ウィンドウ内にPICTとDRAWという2つの形式で受け渡されているのがわかります。

PICTは従来，グラフィックスクリプトと呼ばれていたものがリソースになったもので，ラインやレクタングルといったグラフマンが扱うすべての形式の基本図形の組み合わせを手続きとして記録したものです。

グラフィックスクリプトを作成するのは簡単で，あらかじめ「スクリプトに記述する」というスイッチを入れて描画命令を実行してやれば，その間に行われた画面への描画はすべて（一部例外はあるが）手続きとして記録されます。

一方DRAWは，もともとのグラフマンが備えていなかった機能（内部塗りつぶしのベジエ曲線など）を拡張してあるデータ形式です。現在ではEasydrawとシャーペンのみがサポートしており，PICTに比べて高品位な出力が可能になっています。

といってもSX-WINDOWがver.3.1になったときにグラフマンが拡張されたわけではなく，それぞれのツールが個別に処理をしているだけに留まっています。

ですから，ユーザーレベルで見るとPICTを扱うのは比較的簡単ですが，DRAWを扱うことはかなり難しいといわざるをえないでしょう。ユーザーレベルでドローツールを作成する場合などはPICTにあわせるしかありませんから，どうしても出力の品位は落ちてしまいそうです。

こういった個別のツールによる描画処理は「オーナードロー」と呼ばれています。システムに飽き足らないときに使われるものですが，なにもシャープ製品でやることも……。

ドローデータ

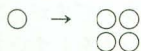
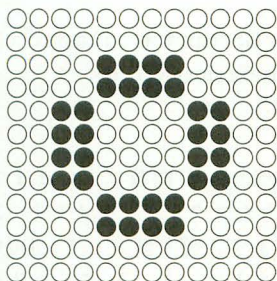
コンピュータは絵を点の集まりで表示します。円や直線など，なめらかそうに見える図形も，すべてディスプレイ上では点の集まりで表されています。

たとえば，「A」という文字は

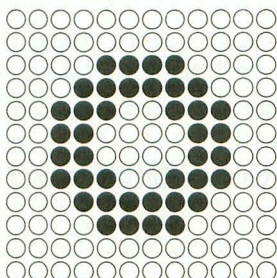
というふうに，表示します。

半径1の円を表示する場合は，

とします。

で，これを2倍する場合を考えてください。

という変換を行うと，

となります。

よくよく考えると，半径1の円の大きさを2倍すると，半径2の円になるわけです。

だったら，

のほうが円に見えます（もしよろしければ，目を1mくらい離して見てください）。

なぜ，このような違いが起こってしまったのでしょう。

確かに，コンピュータの扱う図形は最終的にすべて点の集まりとして処理されます。しかし，それはあくまでも最終的な話であって，中間過程では，なにも点にこだわる必要はありません。

1 → 円
2 → 直線
:

というような，約束事を決めておけば，ひととおりの図形はすべて数値データで図形を表せるわけです。この図形に，拡大/縮小などの作業を加えるときには，点の集まりを伸ばしたり，縮めたりするのではなく，半径を表す数値や座標を表す数値に掛け算/割り算を施します。

中間過程では一定の約束事に従って図形データを扱い，最終的な段階になって，はじめて点の集まりへと変換する。このような方法をとれば，出力時点での最適な変換方法を選択できるわけす。

このように，最終的な段階で初めて点の集まりへと変換する方式をドロー方式といいます。

リスト1

```
 1:
 2: list 2:
 3: /*
 4:  *       グラフスクリプトコマンド (gScript command)
 5:  */
 6: #define GS_NOP          0
 7: #define GS_REM          1
 8: #define GS_BITMAP       2
 9: #define GS_APAGE        3
10: #define GS_CLIP         4
11: #define GS_PMODE        5
12: #define GS_PSIZE        6
13: #define GS_FKIND        7
14: #define GS_FFACE        8
15: #define GS_FMODE        9
16: #define GS_FSIZE        10
17: #define GS_FORE         11
18: #define GS_BACK         12
19: #define GS_PPAT         13
20: #define GS_EXPAT        14
21: #define GS_LINE         15
22: #define GS_FRRECT       16
23: #define GS_FLRECT       17
24: #define GS_FRRRECT      18
25: #define GS_FLRRECT      19
26: #define GS_FROVAL       20
27: #define GS_FLOVAL       21
28: #define GS_FRARC        22
29: #define GS_FLARC        23
30: #define GS_FRPOLY       24
31: #define GS_FLPOLY       25
32: #define GS_FRRGN        26
33: #define GS_FLRGN        27
34: #define GS_STR          28
35: #define GS_PUT          29
36: #define GS_COPY         30
37: #define GS_FRNPOLY      31
38: #define GS_FLNPOLY      32
39:
40: #define GS_END          -1
```

図1　DRAW型で出力

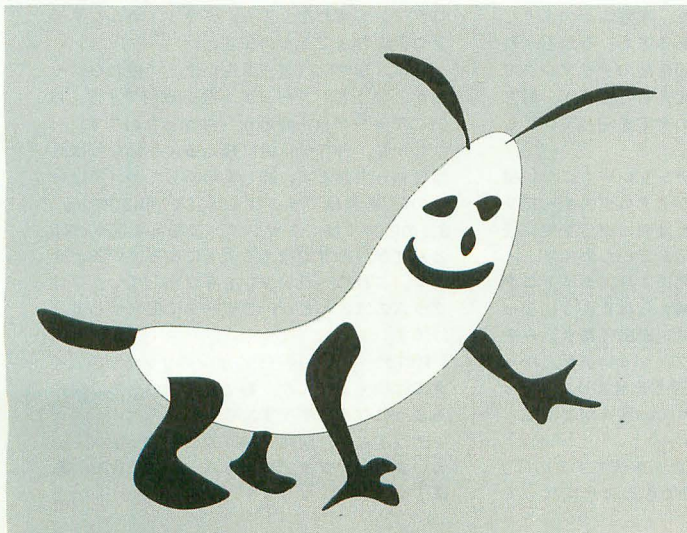
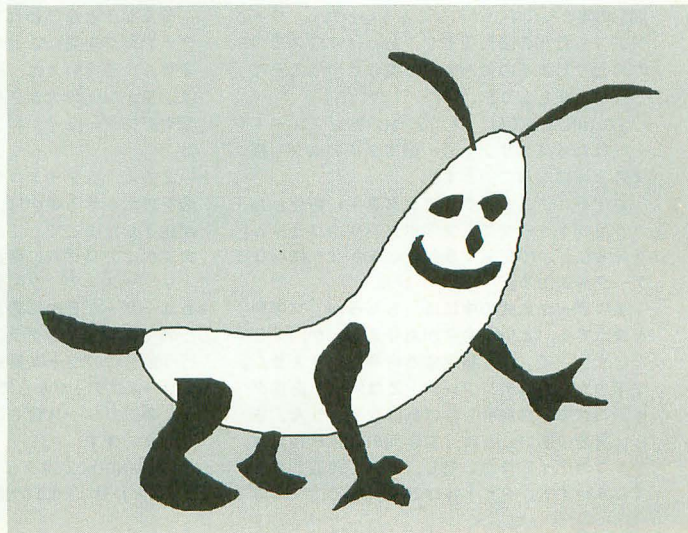
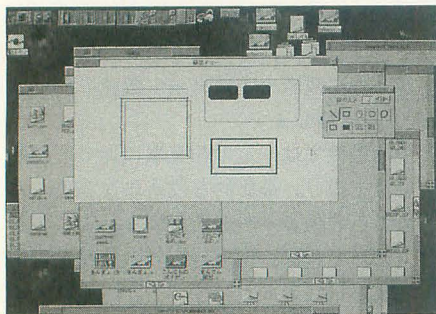



図2　PICT型で出力

改造版簡易ドロー

PICTデータの作成(その1)

現在のところ，SX-WINDOW上でPICTデータの作成を行えるのはEasydrawしかありません。

「Easydraw」は確かに便利なドローツールです。SX-WINDOW上で本格的にグラフィックを扱おうとする人は持っていて，損はないでしょう。

しかし，ちょっと試してみようかな，という感じで買える値段でもありません。

PICTデータの作成(その2)

新たにドローツールを作るのは，たいへん難しいことです。なるべく，既存のものを改造して間にあわせたいものです。

幸い，開発キットの中には「サンプルプログラム応用編」として「簡易ドロー」がソースリスト込みで収録されています。

このプログラムは，C言語でSX-WINDOWのプログラムを作れる人なら，皆持っているわけで，入手についても問題ありません。SX-BASICのウィンドウデザイナを作る際にも，かなりの部分を流用させてもらったので，だいたいの構造はわかっています。扱える図形の種類に若干の不満はありますが，どうしても豊富な図形を扱いたくなったら，それは,「簡易ドロー」ではなく「Easydraw」を使う段階に入ったと考えることにしましょう。

現在のところ，PICT形式のデータをファイルに出力しても，読み込むソフトがないのであまり意味がありません。PICTのデータは主にクリップボード経由でやり取りします。

ただし，

簡易ドロー → クリップボード

は特に問題ないのですが，

クリップボード → 簡易ドロー

は今回はサポートしていません。前述のように，簡易ドローで扱える図形の種類に限りがあり，とても，PICTのすべての図形を読み込むというわけにはいかないからです。また，グラフマンによって，描かれるべき図形を横取りし，PICTに落とす，という動作はSXコール一発で簡単にできるのですが，その逆は1つひとつのデータに対し，自分で変換ルーチンを持たなくてはならないため，かなりやっかいな作業になります。

さて，実際の改造箇所ですが，

1) ユーザーから，クリップボードへの「コピー」命令を受け付ける

2) 実際に，クリップボードへ図形の「コピー」を行う

の2箇所となります。

1)は，メニューかショートカットキーOPT.1+Cで指定しますので，それぞれ，ライトダウンイベントとキーダウンイベントに対応する部分を書き換えます。

2)は，グラフマンで出力先をグラフポートから，スクリプトへ切り替える命令がありますので，これを使用後，通常通りに描画を行い，得られたデータをクリップボードへ送りつけています。

特に，問題はないと思うのですが，ひとつだけ，トリッキーな使い方をしている場所があります。

```
{
    register GScript * * _a0 asm("a0");
    GMCloseScript(NULL);
    gscHdl = _a0;
}
```

GMCloseScript()関数は，いままで描画を行ってきたスクリプトをメモリハンドルに代入します。戻り値として，D0レジスタにリザルトコード，A0にメモリハンドルを返します。C言語では，戻り値としてD0レジスタの値しか読み込めないので，これでは，どのメモリハンドルに書き込まれたのかわかりません。最近のgccではうまい逃げ道があるようですが，ちょっと使い方がわからなかったので，上記のように，_a0をレジスタ変数として宣言し，それを普通の変数のように扱うことにしました。

コンパイル&動作チェック

私は，以下のような環境でコンパイルを行いました。

ダイアログの表示について

SX-WINDOWのダイアログは「ぴんぽーん」と鳴ったりするのを除けば，結局はいろいろな図形の集まりです（そんなことをいったら，ウィンドウもただの図形ですが，あれはマウスでつかめたりしますのでちょっと違うということにしておきましょう）。

SX-WINDOWでは，ダイアログテンプレートと呼ばれるスクリプトの一種でダイアログも扱えるようになっています。

さすがに，これはダイアログ専用に特化したスクリプトデータで，文字ボタンやボリュームを命令ひとつで配置できたりとなかなか便利なデータ形式です。

そのデータ形式の中には，もちろん「文字列を表示する」という命令があります。が，「どんな大きさで」かを指定する命令はありません。文字の大きさはテンプレートとは別に，直接ダイアログマンに命令しなければなりません。さらに具合の悪いことに，文字列の大きさ指定は，ひとつのダイアログに対し，すべて共通となってしまいます。タイトルは24ドットで，文章は12ドットで，というような使い分けができません。

そんなことをいわれても，実際にそのようなダイアログを見たことがある方もいるかもしれません。そういえば，画面暗前のダイアログでは，フォントの大きさを変えるどころか，影文字やアンダーライン付きの文字を使っていました。

このようなダイアログテンプレートにない機能を表示させる場合は，ダイアログのオープンが完了したあとで，これとは別にグラフマン経由でダイアログ内に書き込んでいたのです。

リソースエディタの発売により，ダイアログもユーザーが自由に書き換えられるようになったのですが，このグラフマン経由で書き込みを行う部分はリソース形式になっていませんから，リソースエディタからでは手が出せません。せっかくの，リソース化もこれでは意味がなくなってしまいます。

SX-WINDOWではダイアログを作成するのにDITLと合わせてPICTが使用できるようになっています。DITLに比べ，PICTはグラフィック描画に関する自由度が高いですから，なにもダイアログマンを使わなくても，グラフィカルなダイアログを作成できるのです。

では，なぜPICTデータをいままで使わなかったかというと，ミもフタもない理由ですが，PICTのフォーマットがわからなかったのです。

しかし，PICTが自由に扱えるようになれば話は変わってきます。詳しく解析したわけではないので，断言はできませんが，SX-WINDOW ver.3.1のPICTでは，グラフマンで扱える図形のほとんどが，PICTで扱えるようになっているようです。これでグラフマンを呼び出していたプログラムのほとんどがPICTデータに置き換わるわけです。

PICTデータは，現バージョンのリソースエディタでは扱えませんが，おそらく近いうちに扱えるようになるはずです。

そうすれば，DITLとPICTの2本立になってしまいますが，ダイアログをユーザーが自由に編集できるようになります。

gcc version 1.00 Tool#1 Based on 1.42

HAS v2.25

HLK v2.27

おそらく，多少の違いがあっても動作すると思いますが，念のために動作チェックをしておきましょう。

1) 改造した簡易ドローを立ち上げます。単体で立ち上げても，クリップボードの内容が見られなければチェックできないので，SX-WINDOWのデスクトップ上から立ち上げてください。

2) マウスを使って適当に図形を描いてください。どうせ，図形が正しく転送されているかどうかを見るだけですから，本当に「てきとー」で構いません。

3) マウスの右ボタンを押しメニューを表示させます。メニュー項目の中にちゃんと「コピー」という項目があることを確認します。この項目を選択すると，簡易ドロー上に表示されている図形がクリップボードに転送されているはずです。

4) システムアイコン（「終了」とか「再起動」とかを選ぶアイコン）から，「クリップボード」を選択します。すると，専用のウィンドウが表示されるので，先ほどの図形が転送されているかを確認します。

「リスト表示」や「16進表示」でうまく表示されるのに「内容表示」で真っ白になってしまう場合は，クリップボードのウィンドウを大きくするか，メニューから「ウィンドウサイズ」を選択することで，表示されるはずです。

SX-BASICとの関係

編集部にはSX-BASICを使った投稿が寄せられ始めています。わりと軽い気持ちで作り始めた二代目SX-BASICですが，予想以上に多くの読者に使ってもらい，作者としては，うれしい限りです。

こんなに多くの読者に使ってもらっているのに，いつまでも暫定版というわけにもいかないでしょう。特に，グラフィック機能がすっぽりと抜け落ちているのは，惜しいところです。

投稿作品を見ていると，この人たちの手元にグラフィック機能付きのSX-BASICがあればなぁ，とついつい考えてしまいます。

そんなわけで，現在，時間を見つけてはSX-BASICのバージョンアップを行っています。おそらく，次回の付録ディスクには間にあうと思うのですが，主にグラフィック部分の強化を行っています。

リスト2 変更点

```
 1: /****************************************************************
 2:  *       selectMenu():    ポップアップメニューの作成と選択処理
 3:  ****************************************************************
 4:  *       引数:    ComVal *pcv        共通変数へのポインタ
 5:  *       戻り値: BOOLEAN            = TRUE:   処理完了
 6:  *                                    = FALSE: 作成失敗（終了）
 7:  */
 8: BOOLEAN selectMenu(ComVal *pcv)
 9: {
10:         int item;
11:         Menu **menuHdl;                             /* メニューハンドル      */
12:
13:         /* メニューを作成する    */
14:         if(pcv->subActiveFlag)
15:                 menuHdl = MNConvert(NULL,
16:                         "ツールボックスを閉じる,"
17:                         "^Cクリップボードへコピー",
18:                         0);
19:         else
20:                 menuHdl = MNConvert(NULL,
21:                         "ツールボックスを開く,"
22:                         "^Cクリップボードへコピー",
23:                         0);
24:         if (menuHdl == NULL) {
25:                 pcv->errorCode = 6;         /* 作成できなかった                */
26:                 return FALSE;               /* 失敗したのでFALSEを返す         */
27:         }
28:         /* メニューの表示と選択を行う */
29:         item = MNSelect(menuHdl, pcv->event.ev.where.x_y);
30:         MMHdlDispose(menuHdl);                      /* メニューハンドルを解放する    */
31:         if (item == 1 && pcv->subwinPtr != NULL) {
32:                 if (pcv->subActiveFlag) {
33:                         /* サブウィンドウを非表示にする */
34:                         WSDelist(pcv->subwinPtr);
35:                         pcv->subActiveFlag = FALSE;
36:                 } else {
37:                         /* サブウィンドウを表示する */
38:                         WSEnlist(pcv->subwinPtr);
39:                         pcv->subActiveFlag = TRUE;
40:                 }
41:         } else if(item == 2) ClipCopy(pcv);
42:         return TRUE;                                /* 処理が完了したのでTRUEを返す */
43: }
44:
45: /*
46: ** クリップボードへコピー
47: */
48: void
49: ClipCopy(ComVal *pcv)
50: {
51:         static Rect winSize = { 0, 0, WIN_H, WIN_V }; /* ウィンドウサイズ */
52:         Region  **rgnHdl;
53:         GScript **gscHdl;
54:         void    **cellHdl;
55:         int             *p1, size;
56:
57:         /* グラフポートを自分に設定する */
58:         GMSetGraph(&pcv->windowPtr->graph);
59:
60:         rgnHdl = GMNewRgn();
61:         GMRectRgn(rgnHdl, &winSize);
62:         GMOpenScript(rgnHdl);
63:         drawnext(pcv, pcv->data);
64:         {
65:                 register GScript **_a0 asm("a0");
66:                 GMCloseScript(NULL);
67:                 gscHdl = _a0;
68:         }
69:         size = MMHdlSizeGet(gscHdl);
70:         cellHdl = MMChHdlNew((size) + 4 + 4);
71:         p1 = (int *)*cellHdl;
72:         p1[0] = 'PICT';
73:         p1[1] = size;
74:         memcpy(&p1[2], *gscHdl, size);
75:         TSPutScrap(size + 4 + 4, cellHdl);
76:         GMDisposeRgn(rgnHdl);
77:         GMDisposeScript(gscHdl);
78: }
79:
80: /****************************************************************
81:  *       keyDownEvent(): キーダウンイベント処理
82:  ****************************************************************
83:  *       引数:    ComVal *pcv        共通変数へのポインタ
84:  *       注釈:    OPT.1+'Q'キーでの終了処理を行う。
85:  *              OPT.1+'C'キーでのクリップボードへのコピーを行う。
86:  */
87: void keyDownEvent(ComVal *pcv)
88: {
89:         int shortCut;                               /* ショートカットキー         */
90:
91:         if (pcv->event.ev.how & KS_OPT1) { /* OPT.1キーが押されたか?    */
92:                 /* キー入力した文字を大文字に変換する */
93:                 shortCut = toupper((int) pcv->event.ev.whom.key.ascii);
94:                 if (shortCut == 'Q')    /* 終了か?                          */
95:                         pcv->endFlag = TRUE; /* 終了フラグをセットする  */
96:                 if (shortCut == 'C')    /* クリップボードへコピー */
97:                         ClipCopy(pcv);
98:         }
99: }
```

並列動作型PICローダ

PICSLICEライブラリ解説

Tan Akihiko 丹 明彦

PICフォーマットの画動ロードをマルチタスクで行うためのローダライブラリです。ロード中にタスクが止まることもなく，おまけに省メモリ。さまざまな用途に使用できます。

PICSLICEはPIC画像ファイルを分割してロードするライブラリである。「稲妻走る」とも呼ばれ分割ロードの難しいPIC画像を数ラスタ単位で少しずつロードでき，メモリ消費量も抑えることができる。SX-WINDOWのような疑似マルチタスクシステムのもとでも動作する。

PICSLICE開発の背景

PICSLICEは，1991年のOh!X付録ディスクに掲載したSXIMAGE.XのPICローダをベースとしている。当時のSX-WINDOWは16色表示のみであり，基本的に16ビットカラーのPICファイルをそのまま表示することができなかった。そのため，いったんメモリ上に16ビットカラーで画像を展開しておき，16色に変換するという戦略で表示していた。このとき，最大512Kバイトの画像バッファをメモリ上に取るのは当時の状況からいって非人道的行為であるという背景から，3ラスタずつ画像をロードして変換するという方式にした。むろん，メモリの空き容量が十分である場合は，高速な一括ローダ（PIC.Rのルーチンをそっくり使っている）を用いる。

3ラスタずつロードすることの困難さであるが，ご存じのとおりPICファイルは「稲妻走る」独特のアルゴリズムによる画像圧縮方式であり，それを3ラスタ切り出すのは容易なことではない。ここでは「稲妻バッファ」または「連鎖バッファ」と呼ばれるバッファを導入して，省メモリなプログラムを作ったのであった。

とはいうものの，16ビットカラーを16色に変換するのは意外に重い処理で，このためひとたびロードすると数10秒もの間イベントが止まってしまうことがSXIMAGEバージョン1.0の最大の欠点となっていた。

そこでSXIMAGEバージョン1.1（未公開）ではタイムスライス型の分割ローディングをするようにしたのである。SX-WINDOWのヌルイベントにせっせと3ラスタロードしては16色変換を行っていたのだ。せっかくだからということで，きばってロードしながら表示もするようにもした。少し処理は遅くなるが，ロードを開始した瞬間に次の作業に移れるので，待たされる感覚は減った。

タイムスライス化に伴い，複数のSXIMAGE.XがPICファイルを同時進行でロードできるように改造した。SX-WINDOWアプリケーション制作の経験をお持ちの方はご承知であろうが，SX-WINDOWでは同一実行ファイルは変数を共有するので，タスク（この場合はロードされる各々の画像）ごとに情報を独立して管理しておく必要があるのである。XVIクラスのマシンだと，数枚の絵を同時にロードしてもそこそこの速度で表示できるのでなかなか壮観な眺めではあった。

SXIMAGE.Xバージョン1.0の発表から3年以上経ついまもSXIMAGE.Xバージョン1.1は発表されていない。機能を欲張ろうとして動作が不安定になってしまったためである。そしてSX-WINDOWがバージョン3になってSXIMAGEは必要なくなったというのが大方の見方であろう，私もそう思う。

＊　＊　＊

さて，このPICSLICEアルゴリズムが再び日の目を見るチャンスを得た。それはSX-WINDOWのデスクトップに画像をロードするのが当たり前になった1993年暮れのことであった。フリーソフト「GRROOT.X」の出現により，デスクトップに65536色の絵を敷けるようになったのだ。そして，中野氏がSX-BASICのプロトタイプを使って「10分ごとにデスクトップの絵を取り換える」プログラムを書いた。このプログラムはなかなかよく動いているのであるが，ひとつの欠点は，10分ごとに数秒ほどイベントが止まってしまうことだった。PICファイルをロードしている間はマウスカーソルが踏切になってしまうのだ。これはいただけないので，PICファイルを数ラスタずつに分解して少しずつロードするPICSLICEがにわかに脚光を浴びてきたというわけである。

PICSLICEの戦略

必要なのは，バックグラウンドでG-RAMまたはメインメモリの所定の領域に少しずつPICファイルからロードした画像を展開する機能である。が，アプリケーションとして仕様をガチガチに決めるといいことがない。フリーソフトウェアのGRROOT.Xは画像ロードにIVM.X（SX-WINDOWのビデオマネージャ）を利用することになっているが，私の勉強不足から，IVMにローダを追加できることは知っていても，ローダの作り方がわからない（公開されてもいない）。よって，どうにでも使えるようにライブラリを作ることにした。

SX-WINDOW上のアプリケーションからもHuman68k の（コマンドライン上の）アプリケーションからも利用したいので，次のように仕様を決める。

・ファイルのオープン/クローズはしない。
・ファイルのアクセスにはDOSコールコンパチのファイルハンドルを用いる。

4枚の画像を並列にロード

これらは，SX-WINDOWとHuman68kではファイルのオープン/クローズの方法が異なるためである。SX-WINDOWはTSOpen ()，Human68kはOPEN () でオープンするのが正式な作法である。クローズも同様。ただしこれらのオープンによって得たファイルハンドルは同じものであり，読み込みのためのアクセスは同じREAD()によって行う。

・スーパーバイザモードのコントロールは行わない。

これは呼び出しのシチュエーションが不定なため。

・ワークエリアはアプリケーション側で確保する。またSX-WINDOWのメモリハンドルは用いない。

これはメモリ確保およびアクセスの方法がSX-WINDOWとHuman68kとで異なるため。また必要なワークエリアのサイズがPIC画像によってまちまちなため。

・1ラスタのバイト数は1024バイト，画像幅，画像幅の偶数化，のいずれか。

これは柔軟性のため。

＊　＊　＊

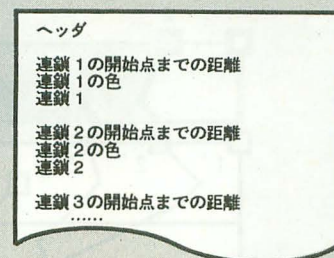
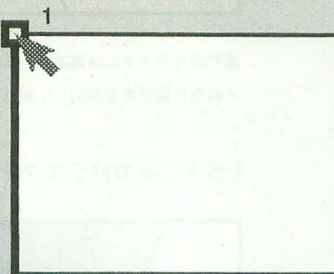
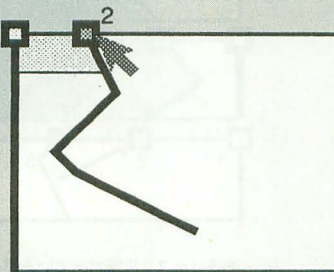
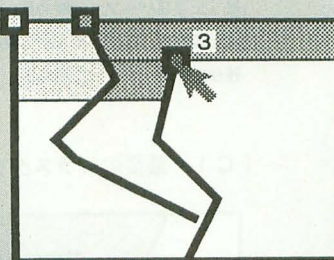
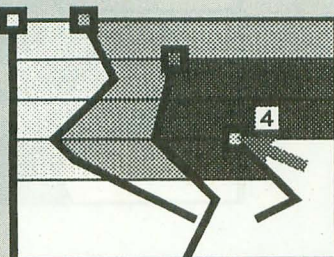
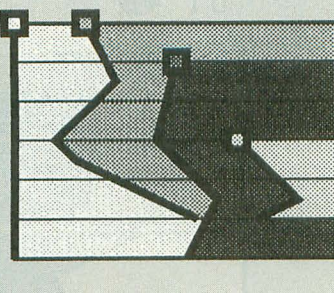
これらの方針に伴ってライブラリマニュアルを作成したので参照していただきたい。

PICSLICEの原理

PICSLICEの真髄は，冒頭にも触れた「稲妻バッファ（連鎖バッファ）」にある。これを理解するためには，PICの画像圧縮/展開方式について，ある程度の予備知識が必要だ。

PICは画像から輪郭を抽出し，稲妻（連鎖）としてファイルに記録する。2次元のデータである画像を2次元的に圧縮するという合理的な方式であり，画像の可逆圧縮方式としてはトップクラスの性能を持っている。しかしその一方で，画像展開時には必ず全画面のデータを一度に格納できる領域が必要である。むろん，直接G-RAMにロードする場合にはなんの問題もないのだが，SX-IMAGEのようにいったんメインメモリ上にロードしておいてから16色に変換するような場合には困ったことになるわけである。1タスクあたり作業エリアに512Kバイトも消費するというのはSX-WINDOWのようなマルチタスクシステムの上で動くアプリケーションとしては望ましくない。

PICの画像展開方式を簡単に解説しておく。PICファイル形式はおおむね次のようになっている。

図1　PICファイルの通常のロード方式

ヘッダ
連鎖1開始点までの距離
連鎖1の色
連鎖1
連鎖2開始点までの距離
連鎖2の色
連鎖2
連鎖3開始点までの距離
⋮
連鎖n開始点までの距離
連鎖nの色
連鎖n

これを展開する様子を図1に示す。展開ポインタは画面左上から右下に向かって順番に移動していく。その途中途中で連鎖データを読み込んでは連鎖を描画する（いわゆる稲妻を走らせる）ことを繰り返していくのである。連鎖と連鎖の間は，展開ポインタが通りすぎたあとを塗りつぶし，連鎖をまたぐたびに塗りつぶし色をその連鎖の色に変える。

＊　＊　＊

そこでPICSLICEである。その基本的な考え方は，「連鎖の刈り取り」にある。関数picsliceLoad()の動作の流れを解説しておこう（図2）。

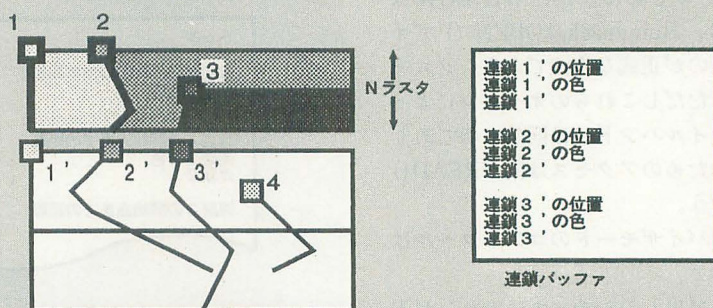
1回分のロード量をNラスタとする。そのNラスタの中に開始点を持つ連鎖データを読み込んで，そのNラスタに入る分だけ連鎖を展開し，残りを刈り取って連鎖バッファに溜める。展開ポインタがNラスタをスキャンし終えたらその回は終了し，関数から戻る。

次回以降は，まず連鎖バッファに溜まっている連鎖をそのNラスタ分だけ展開し，しかるのちにそのNラスタ内に開始点を持つ連鎖を新たに読み込んで，Nラスタだけ展開して残りを刈り取り，連鎖バッファに溜める。この繰り返しである。

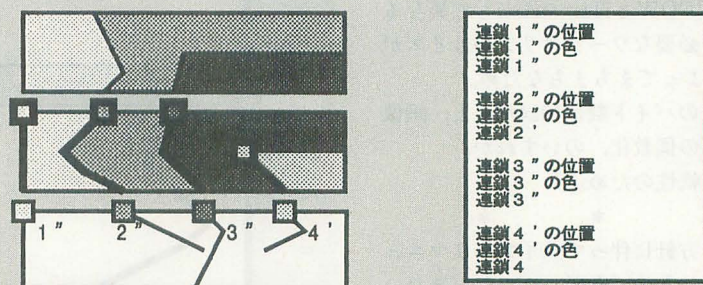
これを実現するための連鎖バッファの構造であるが，おおむねファイルから読み込んだままの形で連鎖を格納しておき，そのうちどこまでを利用したか（Nラスタで刈り取った残り）を指す情報を付加することで，刈り取られた残りを正確に把握できるようにしている。また，どこから連鎖の展開を再開するか（刈り取った点のX座標）という情報も付加するようになっている。限られた連鎖バッファの容量を有効活用するために双方向リストなどを使っている。詳しくはソースコードを読んでほしい。

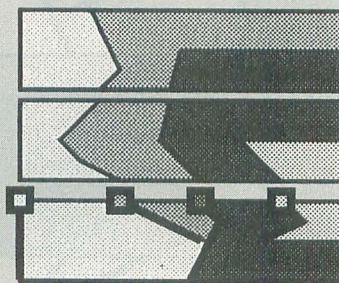
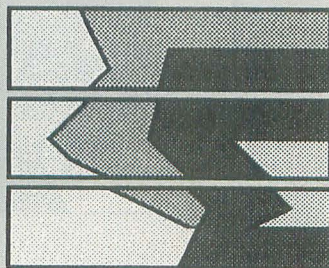
PICSLICEはある条件下で破綻する。それは連鎖バッファがあふれたときである。一般に複雑な画像ほど連鎖バッファがあふ

図2　PICファイルのPICSLICEによるロード方式

（A）　最初のNラスタを展開したところ

最初のNラスタには連鎖1～3が入っている。
それらを途中まで展開し，残りを連鎖バッファに溜める。

（B）　次のNラスタを展開したところ

連鎖1"の位置
連鎖1"の色
連鎖1"
連鎖2"の位置
連鎖2"の色
連鎖2"
連鎖3"の位置
連鎖3"の色
連鎖3"
連鎖4'の位置
連鎖4'の色
連鎖4

連鎖バッファに溜まっている連鎖1'～3'を途中まで展開する。
残りを連鎖バッファに溜める。
新たに連鎖4をファイルから読み込んで展開する。
残りを連鎖バッファに溜める。

（C）　最後のNラスタを展開したところ

（D）　完成

れやすい。特に寿命の長い(縦方向に長い)連鎖バッファが多いとそうなる。こうした事態に対処できるように連鎖バッファの容量は可変になっている。というよりもPICSLICEライブラリは面倒をみない。連鎖バッファはアプリケーション側で確保して，そのサイズをPICSLICEライブラリに伝えるだけである。

したがって，アプリケーション制作者は，複雑な画像を多くロードする可能性があるなら連鎖バッファを大きめにとることを考えておけばよいだろう。サンプルプログラムでは連鎖バッファとして4Kバイトほど確保しているが，経験上これでたいていの画像はロードできる。この倍くらい確保しておけば，実用上問題ないのではないだろうか。

おわりに

PICSLICEを独立したライブラリとしてまとめたのはおよそ半年前だが，基本的にコードを書いたのは3年以上前の話であり，細かい部分はもはや忘却の彼方である。SX-WINDOWが進化したいまとなっては存在価値が薄い気もするが，もしも利用法を見つけ出したならば活用していただけると幸いである。

脱IVM宣言

ご存じのようにSX-WINDOWの画面はテキストとグラフィックという2本の柱で構成されている。テキスト画面の柱は大理石のように美しくしっかりとしているが，グラフィック画面の柱は藁でできているように頼りない。

さて，SX-WINDOWでは65536色のグラフィックはGRWウィンドウ内に表示される。GRWウィンドウの仕様自体はそれなりに妥当なものと評価できる。ウィンドウシステムを統一的にするならオーバーラッピングマルチウィンドウシステムを取り入れざるをえないだろう。

となると手は2つ。デスクトップを512×512ドットに固定するか，グラフィックエリアを設けるかだ。どちらも一長一短。はっきりいえば，どちらも望ましいわけではない。

ハードウェアの制約がある以上，理想論ではすまないところがある。なんといってもG-RAMが1.5Mバイトも足りないのだ。また表示されるデータと同等なものがメモリ上にも必要となるのだが，使用できるメモリが12Mバイトしかない現在，SX-WINDOW上で65536色グラフィックを扱うことはあまり現実的ではないといわざるえない。

グラフィックロード時のタスク占有やメモリの大量消費，色変換の遅さ，ほとんどバグといえる誤差分散の汚さなど，IVMの抱える問題点

仕様がわかれば適切なルーチンに差し換えてやることもできるのだろうが，SX-WINDOW ver.2の資料が公開されたのがこの春で，しかもIFMの資料などはほとんど含まれていなかったことを考えると，望みは薄い。

前例からして，今後，万一グラフィック機能が強化されるようなことがあった場合，現在のシステムとはまた別の管理方法がとられる可能性も高い。

すでに解像度が高ければ16色でも十分な表示能力を持つことが確認されている。

本体改造を伴うとはいえ，宿願のハイレゾ化フル画面表示を達成した現在（無改造でもかなり広げることはできる)，16色モードは脚光を浴びることになるかもしれない。

リスト1

```
 1: /*
 2:  *     picslice.h
 3:  *     - 65536色PICファイルの分割ローダ
 4:  *       Jan. 1994    A.Tan (Oh!X)
 5:  */
 6: 
 7: #ifndef     __PICSLICE_H__
 8: #define     __PICSLICE_H__
 9: 
10: /* PIC分割ロード情報を格納する構造体 */
11: typedef struct _picsliceinfo {
12:     int           filehandle;        /* ファイルハンドル */
13:     unsigned char    *filebuf;       /* ファイルの一時読み込みバッファ */
14:     long           filebufsize;      /* そのサイズ */
15:     int           currentbyte;       /* filebuf 上の読み込み位置 */
16:     int           realwidth;         /* 画像の真の幅 */
17:     int           width;             /* 画像のロード時の幅 */
18:     int           height;            /* 縦方向の長さ */
19:     int           currenty;          /* どこまでロードしたか */
20:     long           runlength;        /* 変化点間の距離およびカウンタ */
21:     unsigned short    color;         /* 変化点間の色 */
22:     unsigned short    *chainbuf;     /* 連鎖バッファ */
23:     int           chainbufsize;      /* 連鎖バッファのサイズ */
24:     long           top;              /* 双方向リストの先頭 */
25:     long           bottom;           /* 双方向リストの末尾 */
26:     long           freelist;         /* フリーリストの先頭 */
27:     int           bytecount;         /* バイト単位のカウンタ */
28:     int           bitcount;          /* ビット単位のカウンタ */
29:     unsigned short    colortable[128];  /* 色履歴リスト */
30:     int           nexttable[128];    /* 次色を示すポインタ */
31:     int           prevtable[128];    /* 前色を示すポインタ */
32:     unsigned short    currentcolor;     /* 現在色 */
33: } picsliceinfo;
34: 
35: /* フレームバッファの横幅の決めかた */
36: #define          PICSLICE_GRAM        101
37: #define          PICSLICE_EXACT       102
38: #define          PICSLICE_EVEN        103
39: 
40: /* リターンコード */
41: #define          PICSLICE_NORMAL      200
42: #define          PICSLICE_TERMINATE   201
43: #define          PICSLICE_EFORMAT     202
44: #define          PICSLICE_EREAD       203
45: #define          PICSLICE_EOVERFLOW   204
46: 
47: /* 公開されている関数 */
48: int    picsliceOpen( picsliceinfo *psi, int mode );
49: int    picsliceLoad( picsliceinfo *psi, unsigned short *frame, int raster );
50: 
51: #endif    /* __PICSLICE_H__ */
```

リスト2

```
 1: /*
 2:  *     picslice.c
 3:  *     - 65536色PICファイルの分割ローダ
 4:  *       Jan. 1994    A.Tan (Oh!X)
 5:  */
 6: 
 7: #define          __DOS_INLINE__
 8: #include     <doslib.h>
 9: #include     "picslice.h"
10: 
11: static unsigned short    chain_cache[128];
12: static int          cache_p, cache_bit_length;
13: static int          readerror;
14: 
15: /* 連鎖バッファをアクセスするためのインデックス */
16: #define          CB_HEAD      0
17: #define          CB_NEXT      0
18: #define          CB_PREV      1
19: #define          CB_COLOR     2
20: #define          CB_X         3
21: #define          CB_SIZE      4
22: #define          CB_OFFSET    5
23: #define          CB_CHAIN     6
24: #define          CB_NIL       0xFFFF
25: 
26: /* バッファのポインタを増加し，必要ならばファイルからバッファに読み込む */
27: void    getNextBuff( picsliceinfo *psi )
28: {
29:     if ( psi->bytecount == 0 ) {
30:         psi->bytecount = READ( psi->filehandle, (UBYTE *)psi->filebuf, psi->filebufsize );
31:         if ( psi->bytecount == 0 ) readerror = 1;
32:         psi->currentbyte = 0;
33:     } else {
34:         psi->currentbyte++;
35:     }
36:     psi->bytecount--;
37:     psi->bitcount = 8;
38: }
39: 
40: /* size で指定されたビット数ぶんの数値を読み出す */
41: long    readBit( int size, picsliceinfo *psi )
42: {
43:     int          i;
44:     unsigned long    a;
45:     unsigned char    *cb, *filebuf = psi->filebuf;
46: 
47:     cb = &filebuf[ psi->currentbyte ];
48:     a = 0;
49:     while ( size > psi->bitcount ) {
50:         for ( i=0; i < psi->bitcount; i++ ) {
51:             a *= 2;
52:             if ( *cb & 0x80 ) a++;
53:             *cb *= 2;
54:             size--;
55:         }
56:         getNextBuff( psi );
57:         cb = &filebuf[ psi->currentbyte ];
58:     }
59:     for ( i = 0; i < size; i++ ) {
60:         a *= 2;
61:         if ( *cb & 0x80 ) a++;
62:         *cb *= 2;
63:         psi->bitcount--;
64:     }
65:     return a;
66: }
67: 
68: /* 次の色変化点を読み出す */
69: long    readLen( picsliceinfo *psi )
70: {
71:     int    a;
72: 
73:     a = 1;
74:     while ( readBit( 1, psi ) != 0 ) a++;
75: 
76:     return ( readBit( a, psi ) + (1 << a) - 1 );
77: }
78: 
79: /* 連鎖キャッシュの書き込み */
80: void    writeCache( int size, short n, picsliceinfo *psi )
81: {
82:     int    i;
```

```
 83:
 84:     n <<= (16-size);
 85:     for ( i = 0; i < size; i++ ) {
 86:         chain_cache[cache_p] = chain_cache[cache_p]*2 + (n<0);
 87:         n = n + n;
 88:         if ( --cache_bit_length == 0 ) {
 89:             cache_p++;
 90:             cache_bit_length = 16;
 91:         }
 92:     }
 93: }
 94:
 95: /* 連鎖バッファのガーベジコレクション */
 96: void    garbageCollection( picsliceinfo *psi )
 97: {
 98:     unsigned short    offset, p , prev, newp;
 99:     int         i, l;
100:     unsigned short    *chain = psi->chainbuf;
101:
102:     if ( psi->top == CB_NIL ) {
103:         psi->top = psi->bottom = CB_NIL;
104:         psi->freelist = 0;
105:         return;
106:     }
107:     for ( p = psi->top; p != CB_NIL; p = chain[ p + CB_NEXT ] ) {
108:         offset = chain[ p + CB_OFFSET ];
109:         chain[ p + CB_OFFSET ] = offset%16;
110:         offset /= 16;
111:         for ( i = 0; i < (chain[ p + CB_SIZE ] - (CB_CHAIN-CB_NEXT) - offset); i++ )
112:             chain[ p + CB_CHAIN + i ] = chain[ p + CB_CHAIN + offset + i ];
113:         chain[ p + CB_SIZE ] -= offset;
114:     }
115:     if ( psi->top > 0 ) {
116:         if ( chain[ psi->top + CB_NEXT ] != CB_NIL ) {
117:             chain[ chain[ psi->top + CB_NEXT ] + CB_PREV ] = 0;
118:             chain[ 0 + CB_NEXT ] = chain[ psi->top + CB_NEXT ];
119:         } else {
120:             chain[ 0 + CB_NEXT ] = CB_NIL;
121:             psi->bottom = 0;
122:             psi->freelist = 0 + chain[ psi->top + CB_SIZE ];
123:         }
124:         l = chain[ psi->top + CB_SIZE ];
125:         for ( i = CB_PREV; i < l; i++ ) {
126:             chain[ 0 + i ] = chain[ psi->top + i ];
127:         }
128:         psi->top = 0;
129:     }
130:     if ( chain[ psi->top + CB_NEXT ] != CB_NIL ) {
131:         for ( p = chain[ psi->top + CB_NEXT ]; p != CB_NIL; p = chain[ p + CB_NEXT ] ) {
132:             prev = chain[ p + CB_PREV ];
133:             newp = prev+chain[ prev + CB_SIZE ];
134:             if ( newp < p ) {
135:                 l = chain[ p + CB_SIZE ];
136:                 for ( i = 0; i < l; i++ ) {
137:                     chain[ newp + i ] = chain[ p + i ];
138:                 }
139:                 chain[ prev + CB_NEXT ] = newp;
140:                 if ( chain[ newp + CB_NEXT ] == CB_NIL ) {
141:                     psi->freelist = newp + chain[ newp + CB_SIZE ];
142:                     psi->bottom = newp;
143:                 } else {
144:                     chain[ chain[ newp + CB_NEXT ] + CB_PREV ] = newp;
145:                 }
146:                 p = newp;
147:             }
148:         }
149:     }
150:     return;
151: }
152:
153: /* 連鎖を指定ラスタ数ぶん展開する */
154: int    expandChain( unsigned short *frame, int x, int y0, int ry, unsigned short c, picsli
einfo *psi, int raster )
155: {
156:     unsigned short    l;
157:     int         i, y;
158:
159:     y = y0 + ry;
160:     for (;;) {
161:         switch ( readBit( 2, psi ) ) {
162:         case 0:    if ( readBit( 1, psi ) == 0 ) return PICSLICE_NORMAL;
163:             if ( readBit( 1, psi ) == 0 ) x -= 2;
164:                 else x += 2;
165:             break;
166:         case 1:    x--;
167:             break;
168:         case 2:    break;
169:         case 3:    x++;
170:             break;
171:         }
172:         y++;
173:         if ( x < 0 || x >= psi->width || y >= psi->height ) {
174:             /* 残りの連鎖を空読み */
175:             for (;;) {
176:                 while ( readBit( 2, psi ) != 0 );
177:                 if ( readBit( 1, psi ) == 0 ) break;
178:                 readBit( 1, psi );
179:             }
180:             return PICSLICE_NORMAL;
181:         }
182:         if ( (y - y0) == raster ) break;    /* 残りは連鎖バッファへ */
183:         frame[ (y - y0)*(psi->realwidth) + x ] = c;
184:     }
185:
186:     if ( readerror != 0 ) return PICSLICE_EREAD;
187:
188:     /* 連鎖バッファ生成 */
189:     cache_p = 0;
190:     cache_bit_length = 16;
191:     for (;;) {
192:         writeCache( 2, l = (short)readBit( 2, psi ), psi );
193:         if ( l == 0 ) {
194:             writeCache( 1, l=(short)readBit( 1, psi ), psi );
195:             if ( l == 0 ) break;
196:             writeCache( 1, (short)readBit( 1, psi ), psi );
197:         }
198:     }
199:     writeCache( 16, (short)0, psi );
200:
201:     if ( readerror != 0 ) return PICSLICE_EREAD;
202:
203:     if ( ( psi->freelist + cache_p + (CB_CHAIN-CB_HEAD) ) > psi->chainbufsize ) {
204:         garbageCollection( psi );
205:         if ( ( psi->freelist + cache_p + (CB_CHAIN-CB_HEAD) ) > psi->chainbufsize ) {
206:             /* 連鎖バッファあふれ(絵柄が複雑すぎる) */
207:             return PICSLICE_EOVERFLOW;
208:         }
209:     }
210:     psi->chainbuf[ psi->freelist + CB_NEXT ] = CB_NIL;
211:     if ( psi->bottom != CB_NIL ) {
212:         psi->chainbuf[ psi->bottom   + CB_NEXT] = psi->freelist;
213:         psi->chainbuf[ psi->freelist + CB_PREV] = psi->bottom;
214:     } else {
215:         psi->chainbuf[ psi->freelist + CB_PREV] = CB_NIL;
216:     }
217:     psi->chainbuf[ psi->freelist + CB_COLOR  ] = c;
218:     psi->chainbuf[ psi->freelist + CB_X      ] = x;
219:     psi->chainbuf[ psi->freelist + CB_SIZE   ] = cache_p+(CB_CHAIN-CB_HEAD);
220:     psi->chainbuf[ psi->freelist + CB_OFFSET ] = 0;
221:     for ( i = 0; i < cache_p; i++ ) {
222:         psi->chainbuf[ psi->freelist + CB_CHAIN + i ] = chain_cache[i];
223:     }
224:     if ( psi->top == CB_NIL ) psi->top = psi->freelist;
225:     psi->bottom = psi->freelist;
226:     psi->freelist += cache_p + (CB_CHAIN - CB_HEAD);
227:     return PICSLICE_NORMAL;
228: }
229:
230: /* 連鎖バッファから1本の稲妻を削除する */
231: void    disposeChainbuf( long p, picsliceinfo *psi )
232: {
233:     int         next, prev;
234:     unsigned short    *chain = psi->chainbuf;
235:
236:     next = chain[ p + CB_NEXT ];
237:     prev = chain[ p + CB_PREV ];
238:
239:     if ( next == CB_NIL && prev == CB_NIL ) {
240:         psi->top = psi->bottom = CB_NIL;
241:         psi->freelist = 0;
242:         return;
243:     }
244:     if ( prev == CB_NIL ) {
245:         psi->top = next;
246:         chain[ psi->top + CB_PREV ] = CB_NIL;
247:         return;
248:     }
249:     if ( next == CB_NIL ) {
250:         psi->bottom = prev;
251:         chain[ psi->bottom + CB_NEXT ] = CB_NIL;
252:         psi->freelist = psi->bottom + chain[ psi->bottom + CB_SIZE ];
253:         return;
254:     }
255:     chain[ next + CB_PREV ] = prev;
256:     chain[ prev + CB_NEXT ] = next;
257: }
258:
259: /* 連鎖バッファに1本の稲妻を読み込む */
260: unsigned short    readChainbuf( int size, long p, picsliceinfo *psi )
261: {
262:     unsigned short    *chain = psi->chainbuf, a;
263:     int         i;
264:     unsigned int    p1, p2;
265:
266:     p1 = chain[ p + CB_OFFSET ]/16;
267:     p2 = chain[ p + CB_OFFSET ]%16;
268:     a = 0;
269:     for ( i = 0; i < size; i++ ) {
270:         a *= 2;
271:         if ( chain[ p + CB_CHAIN + p1 ]&0x8000 ) a++;
272:         chain[ p + CB_CHAIN + p1 ] *= 2;
273:         if ( ++p2 == 16 ) {
274:             p1++;
275:             p2 = 0;
276:         }
277:     }
278:     chain[ p + CB_OFFSET ] = p1*16+p2;
279:     return a;
280: }
281:
282: /* 連鎖バッファの内容を指定ラスタ数ぶん展開する */
283: void    expandChainbuf( unsigned short *frame, int y0, picsliceinfo *psi, int raster )
284: {
285:     long        p;
286:     int         x, y;
287:     unsigned short    *chain = psi->chainbuf, c;
288:
289:     for ( p = psi->top; p != CB_NIL; p = chain[ p + CB_NEXT ] ) {
290:         c = chain[ p + CB_COLOR ];
291:         x = chain[ p + CB_X ];
292:         y = y0;
293:         frame[ (y - y0)*(psi->realwidth) + x ] = c;
294:         for (;;) {
295:             switch ( readChainbuf( 2, p, psi ) ) {
296:             case 0:    if ( readChainbuf( 1, p, psi ) == 0 ) {
297:                     disposeChainbuf( p, psi );
298:                     goto NEXTCHAIN;
299:                 }
300:                 if ( readChainbuf( 1, p, psi ) == 0 ) x-=2;
301:                     else x += 2;
302:                 break;
303:             case 1:    x--;
304:                 break;
305:             case 2:    break;
306:             case 3:    x++;
307:                 break;
308:             }
309:             y++;
310:             if ( x < 0 || x >= psi->width || y >= psi->height ) {
311:                 disposeChainbuf( p, psi );
312:                 goto NEXTCHAIN;
313:             }
314:             if ( (y - y0) == raster ) {
315:                 chain[ p + CB_X ] = x;
316:                 goto NEXTCHAIN;
317:             }
318:             frame[ (y - y0)*(psi->realwidth) + x ] = c;
319:         }
320: NEXTCHAIN:
321:         continue;
322:     }
323: }
324:
325: /* 色テーブルを初期化する */
326: void    initColorTable( picsliceinfo *psi )
327: {
328:     int    i;
329:
330:     for ( i=0; i<128; i++ ) {
331:         psi->colortable[i] = 0;
332:         psi->prevtable[i] = i+1;
333:         psi->nexttable[i] = i-1;
334:     }
335:     psi->prevtable[127] = 0;
336:     psi->nexttable[0] = 127;
337:     psi->currentcolor = 0;
338: }
339:
340: /* 新しい色を得る */
341: int    getNewColor( int c, picsliceinfo *psi )
342: {
343:     psi->currentcolor = psi->prevtable[ psi->currentcolor ];
344:     psi->colortable[ psi->currentcolor ] = c;
345:     return ( c*2 );
```


▶「ぷよぷよラムネ」という100円のお菓子があるのを知っていますか？　オマケについてくるぷよ１つと登場キャラ１つの人形がいい味を出しています。着色したら，見栄えがとてもよくなったので部屋に飾ってあります。お勧めです。　　津村　忠蔵(19)佐賀県

```
346: }
347:
348: /* 色を得る */
349: int     getColor( int idx, picsliceinfo *psi )
350: {
351:     int         *nt, *pt;
352:     unsigned short    cc;
353:
354:     nt = psi->nexttable;
355:     pt = psi->prevtable;
356:     cc = psi->currentcolor;
357:
358:     if ( cc != idx ) {
359:         /* idxを色テーブルから削除 */
360:         nt[ pt[ idx ] ] = nt[ idx ];
361:         pt[ nt[ idx ] ] = pt[ idx ];
362:         /* idxを新しい色テーブルにセットする */
363:         nt[ pt[ cc ] ] = idx;
364:         pt[ idx ] = pt[ cc ];
365:         pt[ cc ] = idx;
366:         nt[ idx ] = cc;
367:         psi->currentcolor = idx;
368:     }
369:     return ( psi->colortable[ idx ] * 2 );
370: }
371:
372: /* カラーコードを読み出す */
373: int     readColor( picsliceinfo *psi )
374: {
375:     if ( readBit( 1, psi ) == 0 )
376:         return ( getNewColor( (int)readBit( 15, psi ), psi ) );
377:     else
378:         return ( getColor( (int)readBit( 7, psi ), psi ) );
379: }
380:
381: /* PICファイルの分割ロードの開始(公開されている関数) */
382: int     picsliceOpen( picsliceinfo *psi, int mode )
383: {
384:     readerror = 0;
385:
386:     /* PIC ファイルの識別 */
387:     if ( (int)FGETC( psi->filehandle ) != 'P' ) return PICSLICE_EFORMAT;
388:     if ( (int)FGETC( psi->filehandle ) != 'I' ) return PICSLICE_EFORMAT;
389:     if ( (int)FGETC( psi->filehandle ) != 'C' ) return PICSLICE_EFORMAT;
390:     /* コメント読み飛ばし */
391:     while ( (int)FGETC( psi->filehandle ) != 26 );
392:     while ( (int)FGETC( psi->filehandle ) != 0 );
393:     /* 画面モードのチェック(現在は0のみ対応) */
394:     if ( (int)FGETC( psi->filehandle ) !=  0 ) return PICSLICE_EFORMAT;
395:     if ( (int)FGETC( psi->filehandle ) !=  0 ) return PICSLICE_EFORMAT;
396:     /* 使用する色のビット数のチェック(現在は15のみ対応) */
397:     if ( (int)FGETC( psi->filehandle ) !=  0 ) return PICSLICE_EFORMAT;
398:     if ( (int)FGETC( psi->filehandle ) != 15 ) return PICSLICE_EFORMAT;
399:
400:     /* psi->filebufはmalloc()か相当品で確保しておくこと */
401:     /* psi->filebufsizeには確保したサイズを代入すること */
402:     psi->currentbyte = 0;
403:     psi->width   = (int)FGETC( psi->filehandle )*256;
404:     psi->width  += (int)FGETC( psi->filehandle );
405:     psi->height  = (int)FGETC( psi->filehandle )*256;
406:     psi->height += (int)FGETC( psi->filehandle );
407:     /* ロード領域の幅 */
408:     switch ( mode ) {
409:     case PICSLICE_GRAM:
410:         psi->realwidth = 512;
411:         break;
412:     case PICSLICE_EXACT:
413:         psi->realwidth = psi->width;
414:         break;
415:     case PICSLICE_EVEN:
416:         psi->realwidth = psi->width + (psi->width % 2);
417:         break;
418:     }
419:     psi->currenty = 0;
420:     psi->color = 0;
421:     /* psi->chainbufはmalloc()か相当品で確保しておくこと */
422:     /* psi->chainbufsizeには確保したサイズを代入すること */
423:     psi->top        = CB_NIL;
424:     psi->bottom     = CB_NIL;
425:     psi->freelist  = 0;
426:     psi->bytecount = 0;
427:     psi->bitcount  = 0;
428:     initColorTable( psi );     /* psi->{colortable,nexttable,prevtable,currentcolor}の初期化
*/
429:     /* 最初の稲妻の開始点を得る(初回のみ1を引く) */
430:     psi->runlength = readLen( psi ) - 1;
431:     if ( readerror != 0 ) return PICSLICE_EREAD;
432:
433:     return PICSLICE_NORMAL;
434: }
435:
436: /* PICファイルの分割ロード(公開されている関数) */
437: int     picsliceLoad( picsliceinfo *psi, unsigned short *frame, int raster )
438: {
439:     int         x, y, xy, x0, y0, dy;
440:     unsigned short    chaincolor;
441:     int         bit;
442:
443:     readerror = 0;
444:     y0 = 0;
445:     x0 = 0;
446:     /* 連鎖バッファから現在のスライス分に含まれる稲妻を吐き出す */
447:     expandChainbuf( frame, psi->currenty, psi, raster );
448:     /* 現在のスライス分に含まれる稲妻の開始点を得る */
449:     /* 開始点間の距離はrunlengthに格納されている */
450:     for (;;) {
451:         /* 初回のrunlengthはpicsliceOpen()で得られている */
452:         while ( (x0 + psi->runlength) >= psi->width ) {
453:             psi->runlength -= (psi->width - x0);
454:             y0++;
455:             x0 = 0;
456:             if ( y0 == raster || (psi->currenty + y0) == psi->height ) {
457:                 /* 次の開始点を探索する間に現在のスライス分を */
458:                 /* 越えるか画像終端に達すると */
459:                 /* 現在のスライス分を塗りつぶす */
460:                 dy = (psi->height - psi->currenty);
461:                 if ( dy > raster ) dy = raster;
462:                 for ( y = 0; y < dy; y++ ) {
463:                     xy = y * (psi->realwidth);
464:                     for ( x = 0; x < psi->width; x++ ) {
465:                         if ( frame[xy] & 1 ) psi->color = (frame[xy] & 0xFFFE);
466:                         frame[xy] = psi->color;
467:                         xy++;
468:                     }
469:                 }
470:                 /* 画像終端に達している場合は終了 */
471:                 if ( (psi->currenty + y0) == psi->height ) {
472:                     psi->currenty = psi->height;
473:                     return PICSLICE_TERMINATE;
474:                 }
475:                 /* 現在のスライス分の終了 */
476:                 psi->currenty += raster;
477:                 return PICSLICE_NORMAL;
478:             }
479:         }
480:         /* 現在のスライス分に変化点が含まれている */
481:         /* (x0,y0)が開始点の座標(y0はスライス分内の相対座標) */
482:         x0 += (psi->runlength);
483:         /* 連鎖の色を得る */
484:         chaincolor = ( readColor( psi )|1 );
485:         if ( readerror != 0 ) return PICSLICE_EREAD;
486:         /* 開始点の着色 */
487:         frame[ y0*(psi->realwidth)+x0 ] = chaincolor;
488:         /* 以降の稲妻があるか */
489:         bit = readBit( 1, psi );
490:         if ( readerror != 0 ) return PICSLICE_EREAD;
491:         /* 以降の稲妻を展開 */
492:         if ( bit == 1 ) {
493:             int     ret;
494:             ret = expandChain( frame, x0, psi->currenty, y0, chaincolor, psi, raster );
495:             if ( ret != PICSLICE_NORMAL ) return ret;
496:         }
497:         /* 次の開始点への距離を得る */
498:         psi->runlength = readLen( psi );
499:         if ( readerror != 0 ) return PICSLICE_EREAD;
500:     }
501: }
```

リスト3

```
 1: /*
 2:  * pl4.c
 3:  * - picsliceライブラリのテストプログラム(Human68k版)
 4:  *    4枚の画像を並列ロードする(256×256ドット65536色PIC)
 5:  *    Jan. 1994 - Jun. 1994    A.Tan (Oh!X)
 6:  */
 7:
 8: #define             __IOCS_INLINE__
 9: #include    <iocslib.h>
10: #define             __DOS_INLINE__
11: #include    <doslib.h>
12: #include    <stdlib.h>
13: #include    "picslice.h"
14:
15: /* ファイルバッファのサイズ(単位:バイト) */
16: #ifndef     FILEBUFSIZE
17: #define     FILEBUFSIZE     2048
18: #endif      /* FILEBUFSIZE */
19:
20: /* 連鎖バッファのサイズ(単位:バイト) */
21: #ifndef     CHAINBUFSIZE
22: #define     CHAINBUFSIZE    4096
23: #endif      /* CHAINBUFSIZE */
24:
25: /* 4枚のPICファイルの情報 */
26: picsliceinfo        psi[4];
27: /* 左上の位置(G-RAM上のバイト単位のオフセット) */
28: int                 offset[4] = { 0, 256*2, 256*512*2, 256*512*2 + 256*2 };
29: /* 分割ロードの一回あたりのラスタ数 */
30: int                 raster[4] = { 1, 3, 4, 2 };
31: /* ロード終了フラグ */
32: int                 status[4] = { 0, 0, 0, 0 };
33:
34: void        main( argc, argv )
35: int         argc;
36: char        *argv[];
37: {
38:     int     i, ret, sp, y;
39:
40:     if ( argc < 5 ) {
41:             PRINT( "usage: PL4 左上.PIC 右上.PIC 左下.PIC 右下.PIC¥n" );
42:             PRINT( "        4枚のPICファイル(256×256ドット,32768色)を並列にロードします。¥
n" );
43:             return;
44:     }
45:     /* スーパバイザモード(G-RAMに直接書き込むため) */
46:     sp = SUPER( 0 );
47:     /* 各PICファイルごとに行う初期設定(4枚ぶん) */
48:     for ( i = 0; i < 4; i++ ) {
49:             /* あらかじめ設定しておくpicsliceinfoのメンバ */
50:             psi[i].filehandle = OPEN( argv[i+1], 0x000 );
51:             psi[i].filebuf = (unsigned char *)malloc( FILEBUFSIZE );
52:             psi[i].filebufsize = FILEBUFSIZE;
53:             psi[i].chainbuf = (unsigned short *)malloc( CHAINBUFSIZE );
54:             psi[i].chainbufsize = CHAINBUFSIZE/sizeof(unsigned short);
55:             /* picsliceローディングの開始 */
56:             ret = picsliceOpen( &psi[i], PICSLICE_GRAM );
57:             switch ( ret ) {
58:             case PICSLICE_NORMAL:
59:                     break;
60:             case PICSLICE_EFORMAT:
61:             case PICSLICE_EREAD:
62:                     goto QUIT;
63:                     break;
64:             }
65:     }
66:     /* 512×512ドット65536色モード */
67:     CRTMOD( 12 );
68:     G_CLR_ON();
69:     /* ローディングのメインループ */
70:     for ( y = 0; y < 512; y ++ ) {
71:             /* 4枚を順番に分割ロード */
72:             for ( i = 0; i < 4; i++ ) {
73:                     if ( status[i] != 0 ) continue;
74:                     ret = picsliceLoad( &psi[i], (unsigned short*)(0xC00000+
offset[i]+y*raster[i]*1024), raster[i] );
75:                     switch ( ret ) {
76:                     case PICSLICE_NORMAL:
77:                             break;
78:                     case PICSLICE_TERMINATE:
79:                     case PICSLICE_EREAD:
80:                     case PICSLICE_EOVERFLOW:
81:                             status[i] = 1;
82:                             break;
83:                     }
84:             }
85:     }
86: QUIT:
87:     /* 後始末(4枚ぶん) */
88:     for ( i = 0; i < 4; i++ ) {
89:             free( psi[i].filebuf );
90:             free( psi[i].chainbuf );
91:             CLOSE( psi[i].filehandle );
92:     }
93:     /* ユーザモード */
94:     SUPER( sp );
95:     return;
96: }
```

▶新機種発売は皆が騒ぎだしてから1～2年後と決まっている(?)ので，いまからお金を貯めておきましょう。　荻野　潤(19)埼玉県

SX-WINDOW上のPICローダ

SX-PICSLICE

Ishigami Tatsuya 石上 達也

PICSLICEライブラリを実際にSX-WINDOWで使用するためのツールです。直接G-RAMに描画しますので，グラフィックを使用するアプリケーションとの混用には支障がある場合があります。注意してください。

というわけで，ハードスケジュールをバリバリこなす丹氏が作成したPICSLICEを，わりかし暇だと誤解されている私がSX-WINDOW上でも使えるようにしました。

これで，SX-WINDOWの壁紙として65535色のグラフィックをつけられるようになるわけです。

今回のプログラムと同様の働きをするプログラムには，すでにGRROOT.X(箕浦真氏制作）というのがあります。

詳しくは知りませんが，このGRROOT.XはSX-WINDOWのビデオマネージャを使って与えられた画像ファイルの再生を専用のビッツ*に対して行っているようです。ビッツへの描写が完了したところで，データをグラフィック画面へコピーしているので，展開時にはグラフィック画面とは別に512Kバイトのメモリが必要となります。

また，この描画は一気に行われます。ほとんどの画像データは圧縮された形で保存されていて，描画時には復元して表示されます。この復元作業というのは，わりあい時間のかかる処理です。

SX-WINDOWはひとつしかないCPUを皆でなかよく分けあってマルチタスクを実現しています。時間のかかる処理を誰かが行えばほかのタスクは動作が止まってしまいます。

時間のかかる作業は本来なら小刻みに少しずつ小分けに実行していくべきなのですが，どうやらビデオマネージャにはそのような機能はないようです。

SXPICS.Xはビデオマネージャを使用する代わりに丹氏の作成したPICSLICEライブラリを使用することによって，小刻みな再生が可能となっており，他タスクへの影響が少なくなっています。

また，ビッツを使用せずに直接グラフィック画面に書き込みを行っていますので，画像データの展開時に作業用のメモリをあまり使用しません。

まわりを止めずに画像をロード

*ビッツ
SX-WINDOW用の仮想画面の一種です。詳しくは，参考文献の2-54を参照。

グラフィック画面の表示

SX-WINDOWでは，コントロールパネルから「背景選択」を実行することにより，壁紙を自由に変更することができます。

ふつうは，方眼模様やタイル模様を壁紙として使用するのですが，PAT4形式のグラフィックデータを使用することができます(パターンエディタで表示→「コピー」メニューでクリップボードへ転送→「背景選択」へペースト)。

で，なぜだかは知らないのですが，ここで，16×16ドットの大きさの透明なパターンを背景として用いると，背景(テキスト画面）が透けてグラフィック画面が表示されるようになります。

使い方

SX-PICSは，起動時にパラメータとして命令を渡す方法と起動後，SX-BASICからタスク間通信によって命令を渡す方法の2通りの動作が可能となっています。

●コマンドラインから

-f ファイル名

与えられたファイル名で示されるファイルを表示します。

例) id=fock(″sxpics.x -fcc.pic″)

-v0

起動後，自分のウィンドウを開きません。

-r 数値

SX-PICSは，一定幅のラスターを処理するごとに制御をSX-WINDOWに返しますが，その幅を指定します。ここに512を指定すると一画面分一気に展開するようになります。

例) id=fock (″ sxpics.x -f10″)

●SX-BASICから

FILE ファイル名

与えられたファイル名で示されるファイルを表示します。

例) int id
　　id=fock (″ sxpics.x″)
　　sendmes (id, ″ FILE cc.pic″)

SHOW

SX-PICSのウィンドウを表示します。すでに表示されている場合はなにも起こりません。

HIDE

SX-PICSのウィンドウを隠します。すでに隠れている場合はなにも起きません。

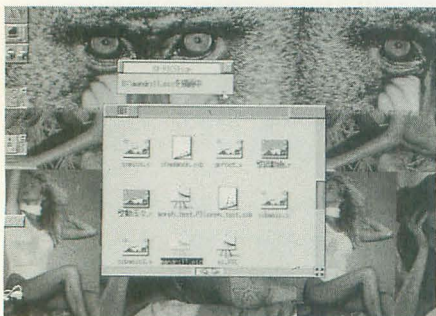
RASTER

SX-PICSは一定幅のラスターを処理するごとに制御をSX-WINDOWに返しますが，その幅を指定します。ここに512を指定すると一画面分一気に展開するようになります。

QUIT もしくは END

SX-PICSを終了します。

複数のSX-PICSを起動しようとした場合

SX-PICSはX68000のグラフィック画面をウィンドウ情報に関係なく1枚の描画領域として扱います。描画領域がひと

つしかないので，SX-PICSが複数立ち上がっても意味がありません（FM音源がひとつしかないのにサウンドプレーヤが複数あっても意味がないのと同様）。

で，2つ目以降のSX-PICSは，

・コマンドラインにパラメータが与えられていなかった場合。
　→直ちに終了。

・コマンドラインにパラメータが与えられていた場合。
　→すでに起動されているSX-PICSに対し，タスク間通信を用いて，自分に与えられたパラメータを託して，終了。

というふうに動作します。

*　　*　　*

SX-PICSは以下のツールを用いて作成しました。制作者の方々に感謝します。

GCC version 1.00 Tool#1 Based on 1.42
HAS v2.25
HLK v2.27

また，SX-WINDOWの背景を透明にする方法に関しては，GRROOT.X（箕浦真氏作成）を参考にしました。

参考文献

SX-WINDOW開発キットプログラマーズマニュアル，シャープ

背景切り換えプログラム

背景を自動的に切り換えるプログラムをSX-PICS対応としてみました。もちろんSX-BASICから起動します。

指定したPICファイルをランダムに切り換えます。特にコンフィグファイルなどは用意していませんので，プログラム中に直接，絶対パス指定で記述してください。デフォルト設定では90秒に1回背景を切り換えます。ロード中は多少処理が重くなるものの，きちんとマルチタスクで動作しますので作業にはほとんど影響を与えないでしょう。

今回は時間を計測するのにSX-BASICのタイマ割り込みを使用してみました。この割り込みはウィンドウエンジンが発生しているものなので，SX-BASIC本体側で動いているプログラムをBREAKしても止まりません。止めたいときはウィンドウエンジンを落としてください。

機能を解説します。このプログラムのウィンドウ内をクリックするか，キーボードのコード入力だけが押されているときには画面上のテキスト画面を非表示にします。全面グラフィック画面になりますのでデータの確認などに便利です。コントロールキーとの併用を推奨します。SADJUST.Rをお使いの方は画面モードも変更するようにするといっそうよいでしょう。

また，壁紙動画にも対応しました。ファイル指定で壁紙動画の起動パラメータが指定してあるときには画像ロード後にそのパラメータに従って背景を動かします。

```
 1: ▼Window Size (100,48),0,0,Untitled
 2: int x,y,i,k,p(50),w,z,r,ff(32),l,L1,L0
 3: str f(99)[64]
 4: str f2[64],t[32]
 5: int a,b,c,e
 6: f(0)="c:¥matier¥ttt.pic" :/* データを作って適当に並べる
 7: f(1)="c:¥matier¥ss1.pic" :/* 動画なら起動オプションも指定
 8: f(2)="d:¥abg¥op01.pic -S40,320 -A220"
 9: f(3)="d:¥abg¥chadance.pic -S16,128 -A240"
10: f(4)="d:¥abg¥morph_test.pic -S136,64 -A200"
                     :
12: f(9)="d:¥abg¥moonec.pic -S40,320 -A220"
13: c=9      :   /* データ総個数－1（手抜き）
14: b=-9999
15: e=findtskn("sxpics.x",0)
16: k=findtskn("壁紙動画.r",0)
17: /*if e=-1 then e=fock("sxpics.x")
18: if k=-1 then k=fock("壁紙動画.r")
19: ▼10,Clock1 (0,0,30,30),0,0,40
20: ▼10,Clock2 (0,0,30,30),0,0,9000
21: /*                            ↑90秒に1回書き換え
22: func Clock1_Timer()
23: if shiftkeybit=1088 or shiftkeybit=1088+8 then pokew(&he82600,31) else poke
w(&he82600,63)
24: endfunc
25: func Clock2_Timer()
26: if k<>-1 then sendmes(k,"STOP"):sendmes(k,"HOME 0,0")
27: a=(rnd()*100000)/1000
28: a=(a*(c+1))/100
29: if b=a then a=(a+1) mod (c+1)
30: f2="FILE "+f(a)
31: L0=len(f2)
32: L1=strcspn(f2,"-")
33: if L0<>L1 then {
34:   t="壁紙動画.r "+right$(f2,L0-L1)
35:   f2=left$(f2,L1)
36:   if k=-1 then k=fock(t) else fock(t)
37: }
38: sendmes(e,f2)
39: if L0<>L1 then for i=0 to 100 :next:sendmes(k,"MOVE")
40: b=a
41: endfunc
42: ▼1,Text1 (0,0,100,48),0,0,0,0,3,0,0,1, Text off
43: func Text1_Click()
44: if k<>-1 then sendmes(k,"HOME 0,0")
45: pokew(&he82600,31)
46: repeat
47: until mousel=0
48: pokew(&he82600,63)
49: endfunc
```

リスト1

```
 1: /*************************************
 2:  * sx.c:   PicSlice For SX-WINDOW
 3:  *         Programmed By ISHIGAMI Tatsuya
 4:  *                                  05/06/94
 5:  *************************************
 6: */
 7:
 8: #include <stdio.h>
 9: #include <fctype.h>
10: #include <stdlib.h>
11: #include <string.h>
12: #include <doslib.h>
13: #include <sxmemory.h>      /* メモリマンを利用するときに必要         */
14: #include <event.h>         /* イベントマンを利用するときに必要       */
15: #include <sxgraph.h>       /* グラフ系マネージャを利用するときに必要 */
16: #include <window.h>        /* ウィンドウマンを利用するときに必要     */
17: #include <dialog.h>        /* ダイアログマンを利用するときに必要     */
18: #include <text.h>          /* テキストマンを利用するときに必要       */
19: #include <task.h>          /* タスクマンを利用するときに必要         */
20: #include "picslice.h"      /* picslice固有のヘッダファイル */
21: #include "sxpics.h"        /* このプログラム固有のヘッダファイル     */
22:
23: char _sxkernelcomm[] = SXKERNEL;   /* カーネル起動コマンド            */
24:
25: /*
26:  * グローバル変数
27:  */
28: Window *windowPtr;         /* ウィンドウポインタ   */
29: BOOLEAN activeFlag;        /* アクティブフラグ     */
30: BOOLEAN drawing;           /* 描画中か                     */
31: TsEvent event;             /* イベントレコード     */
32: int         eventMask;      /* イベントマスク       */
33: int         errorCode;      /* エラーコード         */
34: BOOLEAN endFlag;            /* 終了フラグ           */
35: char        filename[90];   /* ファイル名           */
36: void        **dataHdl;      /* データハンドル       */
37: int         y;              /* 描画中のY座標        */
38: int         dy;             /* 一回の描画幅（縦）   */
39: BOOLEAN     visibleFlag;    /* ウィンドウを表示するか */
40: picsliceinfo        psi;
41:
42: /* バッファのサイズ(単位:バイト) */
43: #define     FILEBUFSIZE     2048
44: #define     CHAINBUFSIZE    8196
45:
46: /* 分割ロードの一回あたりのラスタ数 */
47: #define     RASTER  8
48:
49:
50: /*****************************************************************************
51:  * main():          メイン関数
52:  *****************************************************************************
53:  */
54: int
55: main(void)
56: {
57:
58:     if (!init()) {          /* 初期化処理を行う                 */
59:             showErrDialog();        /* エラーダイアログを表示する   */
60:             endFlag = TRUE; /* 終了フラグをセットする           */
61:     } else {
62:             int                 taskID;                 /* タスクID */
```

```
63:             taskID = TSFindTskn("sxpics.x", 0) & 0xffff;
64:             if(taskID && taskID < TSGetID()) { /* すでに起動されたいる場合 */
65:                     if(filename[0]) SendMes(taskID, "FILE %s", filename);
66:                     endFlag = TRUE; /* 終了フラグをセットする      */
67:             } else {
68:                     if(filename[0]) loadFile(filename);
69:             }
70:     }
71:     if(!endFlag && visibleFlag) WMShow(windowPtr);
72:
73:     /* 終了フラグがセットされるまでループする */
74:     while (!endFlag) {
75:             errorCode = 0;  /* エラーコードをクリアする      */
76:             /* タスクマンからイベントを取得する */
77:             TSEventAvail(eventMask, &event);
78:             /* 自分をカレントグラフにする */
79:             GMSetGraph(&windowPtr->graph);
80:             /* 各イベントに対応した処理を行う */
81:             switch (event.ts.what) {
82:             case E_MSLDOWN:         /* マウス左ボタンダウンイベント */
83:                     msLDownEvent();
84:                     break;
85:             case E_KEYDOWN:         /* キーダウンイベント           */
86:                     keyDownEvent();
87:                     break;
88:             case E_IDLE:            /* アイドルイベント             */
89:                     idleEvent();
90:                     break;
91:             case E_UPDATE:          /* アップデートイベント         */
92:                     updateEvent();
93:                     break;
94:             case E_ACTIVATE:        /* アクティベートイベント       */
95:                     activateEvent();
96:                     break;
97:             case E_SYSTEM1:         /* システムイベント             */
98:             case E_SYSTEM2:
99:                     systemEvent();
100:                    break;
101:            }
102:            if (errorCode != 0)     /* エラーが発生したか?          */
103:                    showErrDialog(); /* エラーダイアログを表示する */
104:    }
105:    /* 終了手続きを行う */
106:    endProc((errorCode == 0) ? EXIT_SUCCESS : EXIT_FAILURE);
107:
108:    return 0;
109: }
110:
111: /*****************************************************************************
112:  * init():          初期化処理
113:  *****************************************************************************
114:  * 戻り値: BOOLEAN           = TRUE:  初期化成功
115:  *                           = FALSE: 初期化失敗 (終了)
116:  * 注釈:    共通変数の初期化、アボート処理関数の設定、ウィンドウの作成と
117:  *          表示などを行う。
118:  */
119: BOOLEAN
120: init(void)
121: {
122:    windowPtr = NULL;       /* ウィンドウポインタ           */
123:    activeFlag = FALSE;     /* アクティブフラグ             */
124:    drawing = FALSE;
125:    eventMask = EVENTMASK; /* イベントマスク               */
126:    errorCode = 0;          /* エラーコード                 */
127:    endFlag = FALSE;        /* 終了フラグ                   */
128:    dy = RASTER;
129:    visibleFlag = FALSE;
130:
131:    TSSetAbort(endProc, NULL); /* アボート処理関数を設定する       */
132:
133:    if (LOWWORD(SXVer()) < SXVER) { /* SXシステムのバージョンを取得する */
134:            errorCode = 1;  /* バージョンが古い             */
135:            return FALSE;   /* 失敗したのでFALSEを返す      */
136:    }
137:    dataHdl = MMChHdlNew(1000);
138:
139:    if (!createWindow()) {  /* ウィンドウを作成する         */
140:            errorCode = 2;  /* 作成できなかった             */
141:            return FALSE;   /* 失敗したのでFALSEを返す      */
142:    }
143:
144:    RecovSize();
145:    return TRUE;            /* 成功したのでTRUEを返す       */
146: }
147:
148: /*****************************************************************************
149:  * createWindow(): ウィンドウの作成
150:  *****************************************************************************
151:  * 戻り値: BOOLEAN          = TRUE:  作成成功
152:  *                          = FALSE: 作成失敗 (終了)
153:  */
154: BOOLEAN
155: createWindow(void )
156: {
157:    int paramFlags;                 /* パラメータの有無             */
158:    Rect rc;
159:    Task task;                      /* タスク管理テーブル           */
160:
161:    static Rect winSize = { 0, 0, WIN_H, WIN_V }; /* ウィンドウサイズ */
162:
163:    TSGetTdb(&task, TS_OWN);        /* タスク管理テーブルを取得する */
164:    paramFlags = TSTakeParam(       /* コマンドラインを解析する     */
165:            task.command,           /* コマンドライン               */
166:            &rc,                    /* 指定されたウィンドウ位置     */
167:            NULL,                   /* 最終文字列を格納しない       */
168:            0,                      /* アドレステーブルを保存しない */
169:            NULL,                   /* アドレステーブルを作成しない */
170:            NULL);                  /* ポインタを確保しない         */
171:    if (!(paramFlags & 1)) {        /* ウィンドウ位置の指定があるか? */
172:            /* ない場合、システムからウィンドウの左上座標を取得する */
173:            rc = winSize;
174:            GMSlideRect(&rc, TSGetWindowPos());
175:    }
176:    windowPtr = WMOpen(     /* ウィンドウをオープンする     */
177:            NULL,                   /* ウィンドウポインタを確保する */
178:            &rc,                    /* ウィンドウの表示位置         */
179:            WINTITLE,               /* ウィンドウタイトル           */
180:            FALSE,                  /* オープン時に描画しない       */
181:            WI_STD2 << 4,           /* タイトルの広い標準ウィンドウ */
182:            WM_FOREMOST,            /* 一番前に表示する             */
183:            TRUE,                   /* クローズボタンを使用する     */
184:            TSGetID());             /* 自分のタスクID               */
185:
186:    if (windowPtr == NULL)
187:            return FALSE;           /* 失敗したのでFALSEを返す      */
188:
189:    return TRUE;                    /* 成功したのでTRUEを返す       */
190: }
191:
192: /*****************************************************************************
193:  * msLDownEvent(): マウス左ボタンダウンイベント処理
194:  *****************************************************************************
195:  * 注釈:
196:  * ウィンドウ上でマウスの左ボタンが押された場合の処理を行う。
197:  * この処理でクローズボタンによる終了、ウィンドウの移動が可能となります。
198:  */
199: void
200: msLDownEvent(void )
201: {
202:    int partCode;                   /* ウィンドウのパートコード     */
203:    Window *wTemp;                  /* テンポラリウィンドウポインタ */
204:
205:    /* イベントが自分のウィンドウか? */
206:    if (event.ev.whom.win == windowPtr) {
207:            /* ウィンドウがインアクティブで、OPT.1キーが押されてないか? */
208:            if (!activeFlag && !(event.ev.how & KS_OPT1)) {
209:                    /* ウィンドウをアクティブにする */
210:                    WMSelect(windowPtr);
211:                    /* ボタンが押された場所のパートコードを取得する */
212:                    partCode = WMFind(event.ev.where.x_y, &wTemp);
213:                    /* タイトルバー以外か、左ボタンが離されたか? */
214:                    if (partCode != W_INDRAG || !EMLStill())
215:                            return;
216:            }
217:            /* マウスのボタンが押されている間、ウィンドウの各種処理をシス
218:             * テムに任せて、ボタンが離された場所のパートコードを取得する
219:             */
220:            partCode = SXCallWindM(windowPtr, &event);
221:            if (partCode == W_INCLOSE) /* クローズボタンか?           */
222:                    endFlag = TRUE; /* 終了フラグをセットする       */
223:    }
224: }
225:
226: void
227: idleEvent(void )
228: {
229:    int     ret, sp;
230:
231:    if(drawing == FALSE)    return;
232:
233:    /* スーパバイザモード(G-RAMに直接書き込むため) */
234:    sp = SUPER( 0 );
235:
236:    /* ローディングのメインループ */
237:    ret = picsliceLoad( &psi, (unsigned short*)(0xC00000+y*1024), dy );
238:    y += dy;
239:    switch ( ret ) {
240:    case PICSLICE_NORMAL:
241:            break;
242:    case PICSLICE_TERMINATE:
243:    case PICSLICE_EREAD:
244:    case PICSLICE_EOVERFLOW:
245:            /* 後始末 */
246:            MMPtrDispose( psi.filebuf );
247:            MMPtrDispose( psi.chainbuf );
248:            TSClose(psi.filehandle);
249:            drawing = FALSE;
250:            drawGraph();
251:            break;
252:    }
253:    /* ユーザモード */
254:    SUPER( sp );
255: }
256:
257: /*****************************************************************************
258:  * keyDownEvent(): キーダウンイベント処理
259:  *****************************************************************************
260:  * 注釈:    OPT.1+'Q'キーでの終了処理を行う。
261:  */
262: void
263: keyDownEvent(void )
264: {
265:    int shortCut;                   /* ショートカットキー           */
266:
267:    if (event.ev.how & (KS_OPT1 | KS_XF2)) { /* OPT.1キーが押されたか?     */
268:            /* キー入力した文字を大文字に変換する */
269:            shortCut = toupper((int) event.ev.whom.key.ascii);
270:            if (shortCut == 'Q')    /* 終了か?                      */
271:                    endFlag = TRUE; /* 終了フラグをセットする       */
272:    }
273: }
274:
275: /*****************************************************************************
276:  * updateEvent():   アップデート処理
277:  *****************************************************************************
278:  * 注釈:
279:  * アップデート処理としてウィンドウ内部を描画する。
280:  * この処理を行わないと、自分より下のウィンドウのアップデートやアイドル
281:  * イベントの処理ができなくなります。
282:  */
283: void
284: updateEvent(void )
285: {
286:    /* イベントが自分のウィンドウか? */
287:    if (event.ev.whom.win == windowPtr) {
288:            WMUpdate(windowPtr); /* アップデートを開始する */
```

▶この時期の食当たりには本当に気をつけましょう(当たった人談)。
藤田　康一(23)静岡県

```
289:            drawGraph();            /* ウィンドウ内部を描画する      */
290:            WMUpdtOver(windowPtr); /* アップデートを終了する         */
291:    }
292: }
293:
294: /*****************************************************************************
295:  * drawGraph():    ウィンドウ内部の描画
296:  *****************************************************************************
297:  */
298: void
299: drawGraph(void )
300: {
301:    /* カレントグラフは、すでに設定されている */
302:
303:    /* ペンモードをバックカラーにする */
304:    GMPenMode(G_BACK << 8 | G_PSET);
305:    GMFillRect(&windowPtr->graph.rect); /* ウィンドウ内をクリアする */
306:    if(drawing) {
307:            char    buff[100];
308:            GMMove(0x00080008);
309:            sprintf(buff, "%sを描画中", filename);
310:            GMDrawStrZ(buff);
311:    }
312: }
313:
314: /*****************************************************************************
315:  * activateEvent():         アクティベートイベント処理
316:  *****************************************************************************
317:  * 注釈:   アクティブ、インアクティブによるアクティブフラグの切り替えと、
318:  *           イベントマスクの切り替えを行う。
319:  */
320: void
321: activateEvent(void )
322: {
323:    /* イベントが自分のウィンドウか? */
324:    if (event.ev.whom.win == windowPtr) {
325:            activeFlag = TRUE;      /* アクティブフラグをセットする */
326:            /* アクティブ時のイベントマスクをセットする */
327:            eventMask |= EM_KEYDOWN;
328:    /* イベントが他のウィンドウで、自分がアクティブ状態か? */
329:    } else if (activeFlag) {
330:            activeFlag = FALSE; /* アクティブフラグをクリアする */
331:            /* インアクティブ時のイベントマスクをセットする */
332:            eventMask &= (~EM_KEYDOWN);
333:    }
334: }
335:
336: /*****************************************************************************
337:  * systemEvent():  システムイベント処理
338:  *****************************************************************************
339:  * 注釈:   全ウィンドウのクローズ、全タスクの終了、ウィンドウのセレクトに
340:  *           対応した処理を行う。
341:  */
342: void
343: systemEvent(void )
344: {
345:    char    *p;
346:
347:    switch (event.ts.what2) {        /* イベントの種類は?          */
348:    case CLOSEALL:                   /* 全ウィンドウのクローズ      */
349:    case ENDTSK:                     /* 全タスクの終了              */
350:            endFlag = TRUE;          /* 終了フラグをセットする      */
351:            break;
352:    case WINDOWSELECT:               /* ウィンドウのセレクト        */
353:            WMSelect(windowPtr); /* ウィンドウをアクティブにする */
354:            break;
355:    case DRAGEND:                    /* ドラッグ終了                */
356:            /* イベントが自分のウィンドウか? */
357:            if (event.ev.whom.win == windowPtr)
358:                    dropIcon();     /* アイコンのドロップ処理を行う */
359:            break;
360:    case SAVE:
361:            SaveSize();
362:            break;
363:    case SX_BASIC_SEND:
364:            p = *(char **)(event.ts.whom);
365:            if(!strcmp("SHOW", p) && visibleFlag == FALSE) {
366:                    WMShow(windowPtr);
367:                    visibleFlag = TRUE;
368:            } else if(!strcmp("HIDE", p) && visibleFlag == TRUE) {
369:                    WMHide(windowPtr);
370:                    visibleFlag = FALSE;
371:            } else if(!strncmp("FILE", p, 4)) {
372:                    loadFile(&p[5]);
373:            } else if(!strncmp("RASTER", p, 6)) {
374:                    sscanf(&p[7], "%d", &dy);
375:            } else if(!strncmp("QUIT", p, 4) || !strncmp("END", p, 3)) {
376:                    endFlag = TRUE; /* 終了フラグをセットする        */
377:            }
378:            break;
379:    }
380: }
381:
382:
383: /************************************************************
384: * SaveSize() コマンドラインにウィンドウのサイズと             *
385: *                  ドラッグされたファイル名(1つ)を書き込む  *
386: *
              *
387: * ウィンドウの位置はその後にシステムが書き込む                *
388: ************************************************************/
389: void
390: SaveSize(void )
391: {
392:    Task    taskBuf;
393:
394:    TSGetTdb(&taskBuf, -1);
395:    sprintf((char *)&taskBuf.command[1], "-R%d -V%d -F%s",
396:            dy, visibleFlag, filename);
397:    taskBuf.command[0] = strlen((char *)&taskBuf.command[1]);
398:    TSSetTdb(&taskBuf, -1);
399: }
400:
401: /************************************************************
402: * RecovSize() コマンドラインからファイル名を取り出す         *
403: ************************************************************/
404: void
405: RecovSize(void )
406: {
407:    Task    taskBuf;
408:    char    *p, *next;
409:
410:    filename[0] = 0;
411:
412:    TSGetTdb(&taskBuf, -1);
413:    next = (char *)&taskBuf.command[1];
414:    while(next != NULL) {
415:            p = next;
416:            next = strchr(p ,' ');
417:            if(next != NULL) *next++ = '¥0';
418:            if(*p == '-' || *p == '/') {
419:                    p++;
420:                    switch(toupper(*p++)) {
421:                            case 'F':
422:                                    strcpy(filename, p);
423:                                    break;
424:                            case 'R':
425:                                    sscanf(p, "%d", &dy);
426:                                    break;
427:                            case 'V':
428:                                    sscanf(p, "%d", &visibleFlag);
429:                                    break;
430:                    }
431:            }
432:    }
433: }
434:
435:
436: /*****************************************************************************
437:  * dropIcon():     アイコンのドロップ処理
438:  *****************************************************************************
439:  */
440: void
441: dropIcon(void )
442: {
443:    int errCode, len;
444:    Rect rc;
445:    Drag *dragPtr;                          /* ドラッグポインタ          */
```

リスト2

```
 1: /*****************************************************************************
 2:  * sxpics.h:            SX-PicSlice For SX-WINDOW用ヘッダファイル
 3:  *****************************************************************************
 4:  *
 5:  * 定数定義
 6:  */
 7: /* ウィンドウタイトル */
 8: #define WINTITLE   ((_LASCII) "¥013SX-PICSlice")
 9:
10: /* ウィンドウサイズ */
11: #define WIN_H            200          /* ウィンドウ幅       */
12: #define WIN_V            30           /* ウィンドウ高さ     */
13:
14: #define TSIZE_H          80           /* テキストの表示行数  */
15: #define TSIZE_V          10
16: #define LHEIGHT          12           /* テキストの改行幅(ドット) */
17: #define F_READ           0
18:
19: /* イベントマスク */
20: #define EVENTMASK  (EM_MSLDOWN | EM_KEYDOWN | EM_UPDATE | EM_ACTIVATE | EM_SYSTEM1 | EM_SYS
TEM2 | 1)
21:
22: /* 属性マスク */
23: #define ATTRMASK   (TS_SYSTEM | TS_VOLID | TS_SUBDIR | TS_ARCH)
24:
25: /* メッセージの送信に用いるIDナンバー          */
26: /* 0 ~ 127までと負数はシステムが使用する  */
27: #define    SX_BASIC_SEND   258
28:
29:
30: /*
31:  * 関数プロトタイプ
32:  */
33: /* sx.c */
34: BOOLEAN init(void );
35: BOOLEAN createWindow(void );
36: void msLDownEvent(void );
37: void idleEvent(void );
38: void keyDownEvent(void );
39: void updateEvent(void );
40: void drawGraph(void );
41: void activateEvent(void );
42: void systemEvent(void );
43: void SaveSize(void );
44: void RecovSize(void );
45: void dropIcon(void );
46: void showErrDialog(void );
47: void endProc(int);
48: BOOLEAN loadFile(char *);
49: void SendMes(int, char *, ...);
```

リスト3

```
# SX-PICS For SX-WINDOW作成用のメイクファイル

EXE     = sxpics.x
C_SW    = -O -c -fomit-frame-pointer -fstrength-reduce -Wall
C_OBJS  = sx.o picslice.o
C_LIBS  = sxlib.l clib.l doslib.l gnulib.l floatfnc.l
C_H             = sxpics.h

%.o::   %.c $(C_H)
        gcc $(C_SW) $<

$(EXE): $(C_OBJS)
        lk -l -O $(EXE) $^ $(C_LIBS)
```

▶マリリオンの「ブレイヴ」というCDを買いました。記憶を失った10代の少女のことをテーマにした大叙情詩です。お勧めです。今年はプログレッシブロックがブレイクしますよ。
林 哲司(17)栃木県

SXにおけるグラフィック究極の無駄遣い術

壁紙動画

Fukushima Shota 福嶋 章太

どうもいまひとつ活躍の場がないSX-WINDOWでのグラフィック画面。どうせなら，ここで最大限に「無駄遣い」をしてみましょう。SXデスクトップの背景をアニメーションさせます。

SX-WINDOWにおけるグラフィックVRAMの位置づけは実にあいまいです。16色と6万色という2種類のモードがあり(256色モードというのも内部的にはあるようだ)，多くのグラフィック使用ソフトは，そのどちらかのモードでしか動作しません。しかもその2種類のモードを切り換えるには，再起動という時間のかかる動作を必要とします（ごく一部のユーザーは再起動することなく，モード切り換えをしているらしいが)。このような事情から，SX-WINDOWにおけるグラフィックVRAMとは，単なるおまけ程度にしか考えられていないのが現状です。

そこで，「もっとグラフィックを有効に活用する方法を考えよう」という方向に普通は話が進みそうなのですが，ここではまったく逆の方向に話を進めてしまいます。要するに，「どうせおまけなら，とことん無駄遣いをしよう」てな感じで進めていきます(究極の無駄遣い＝有効活用なのかもしれませんが)。

無駄遣い度

究極の無駄遣いを目指すために，まずは無駄遣いの実例を挙げてみましょう。

実例1)　普段は使用していないが，ぼ〜っとしていると魚が泳ぎはじめる。

この例はグラフィックのみならずスプライトまで使用しているという点でなかなかの無駄遣い度ですが，ぼ〜っとしているときにしか使用されないのがいまいちです。

無駄遣い度3

実例2)　グラフィックウィンドウを開き，CGビジョン.Xで動画を再生し続ける。

高速なマシンを使用したとしても，あまりサイズの大きいアニメーションはできないでしょう。VRAMの一部しか活用していないのが難点です。

無駄遣い度5

実例3)　デスクトップの背景（壁紙ともいう）にしている。

読者からの質問のはがきもたくさんきているという，ブロンドのお姉さんのことです。なかなかの無駄遣い度といえます。

無駄遣い度7

実例4)　デスクトップの背景にして，一定間隔で自動書き換えをしている。

実例3)の発展型です。これなら飽きることも少なく，ほぼ究極の無駄遣いといえます。

無駄遣い度9

究極の無駄遣い

では上の実例をもとに，究極の無駄遣いを目指していきましょう。

まず，背景にするというのはまずまずの無駄遣い度のようです（実例3)。さらに，ずっと同じ絵では飽きてしまいますので，一定間隔で書き換えます（実例4)。しかし，これでは無駄遣い度9止まりです。

そこで考えられるのが，背景を動かすアニメーションです（実例2＋3)。しかし，単純に書き換えで行おうとしても，マシンのスピードが追いつきません。やはり，アニメーションは無理なのでしょうか。

そんなことはありません。たとえ書き換えをまったくしなくとも，アニメーションは可能なのです。

というわけで，ちょっと強引ですが，背景にしたグラフィックをアニメーションさせるという方向で究極の無駄遣いを目指しましょう。

どのように動かすか

グラフィックVRAMを直接書き換えることなくアニメーションを行うには，2通りの方法があります。パレットチェンジとハードウェアスクロールです。このうちパレットチェンジはなかなか有効な手なのですが，元になるデータを用意するのが難しいので，今回はハードウェアスクロールのみを使用します（パレットチェンジは各自の自由研究ということにしてしまいましょう)。

ハードウェアスクロールによるアニメーションを行うには，まずグラフィックVRAMを何分割かして，そこにコマデータをすべて並べる必要があります（書き換えないのだから当たり前)。

図1を見てください。この場合グラフィックを縦横それぞれ8分割しているので，全体で64コマのデータを並べることができます。これを，1番から64番までが順番に表示されるようにハードウェアスクロールすることによってアニメーションさせるわけです。

しかし，ここで問題があります。図1で1番のコマデータの場所に1番から64番のコマを順番に表示した場合，はじめに2番のコマデータが表示されている場所には表1で示すような順番でコマが表示されてしまいます。同様にはじめに1番以外のコマが表示されている場所では思ったとおりの順番（1から64まで順番に）では表示され

図1　データの並べ方その1

G-VRAM

512ドット

1	9						
2	⋮						
3							
4							
5							
6							⋮
7							63
8							64

ません。これでは画面のひと隅でのみちゃんとした動画が行われているにすぎません。
そこで，すべての場所で同じ順番にコマを表示させるように，工夫してやる必要があります。それが図2と，その拡大図の図3です。なぜこれで都合よく順番に表示されるようになるかは，各自考えてもらうとして，とにかくこれで，背景をアニメーションさせるための基本的な考え方はまとまりました（究極の無駄遣いに一歩近づいたというわけです）。

壁紙動画.R

では，実際背景にグラフィックを表示させて，それをアニメーションさせてみましょう。
私が壁紙動画.Rというプログラムを用意したので，それを使います。
まず，テキストの背景を透明にして，グラフィックを見えるようにしなければなりません（フリーウェアでこの辺のことを一遍に行ってくれるものもありますが，ここではそのような便利ソフトを持っていない場合の話をします)。以下のように行ってください。

パターンエディタ.Xを立ち上げます。
アイコンサイズを縦横ともに16に設定します。
全体（16×16）をマスク（いちばん左のパレット）で塗りつぶします。
全選択をしてコピーします。
コントロールパネルから背景設定を立ち上げます。
先ほどコピーしたパターンをペーストします。
すでにテキストの背景は透明になっているはずですから，あとはそれを設定してしまいましょう。
次に，アニメーションデータとなる絵を表示させます。
これは簡単，キャンバス.Xを開いて絵を表示させ，迷わず実寸大モードです。もちろん今月発表されたSXPICS.Xを使ってもかまいません。
さあこれで，背景にしたグラフィックをアニメーションさせる準備が整いました。
あとは壁紙動画.Rを起動するだけです。

壁紙動画.r -S8,64 -A128

背景がアニメーションして見えれば成功です。
コマデータの配置をいろいろ変えることによって，少しずつ違う見え方がしますので，各自試してみてください。

壁紙動画.R 起動スイッチ

-Mn

動画の再生と停止を設定します。n=0で再生，n=1で停止，またn=−1でトグル動作をします。

-Hx0,y0

グラフィックの初期ホーム座標を設定します。x0,y0にホーム座標を指定します。

-Sx1,y1

1フレームごとのスクロール値を設定します。x1,y1に移動量を指定します。

-Am

動画速度を設定します。mはウェイト値でm=0で最速，m=255で停止です。

-Ko

スタート画面に記憶される場合の動作を設定します。o=0で再起動する，o=1で再起動しない，またo=−1でトグル動作をします。

-R

タスクを終了します。

-Dcommandline

commandlineをそのまま実行します。このスイッチは最後に指定してください。

プログラム解説

このプログラムで重要なのは，垂直同期割り込みによるグラフィックのスクロールルーチンと，SXコール$a412e番（テキスト，グラフィック，両方のホーム座標を設

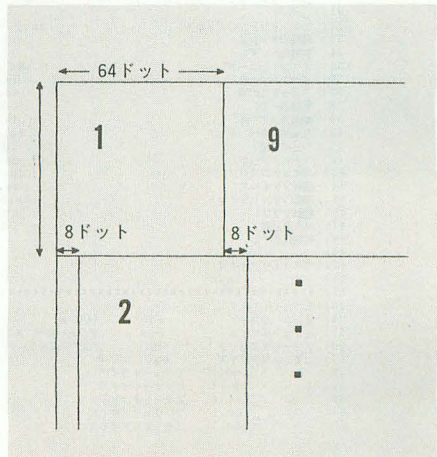
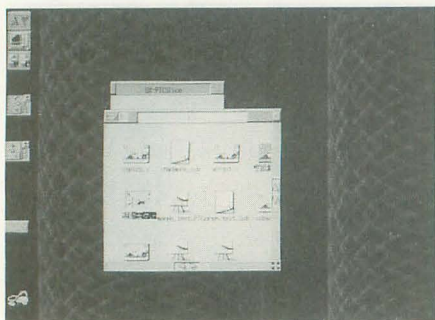
モーフィング画像がうねうね

定するコール）をフックしている部分ですので，その辺を中心に解説します。
まず，初期化ルーチン（305行〜）内で垂直同期割り込みの登録と，$a42eのフックを行っています（343〜359行)。
フック処理は複数のコールを同時にフックできるような構造になっています（壁紙動画.Rではひとつしかフックしていませんが)。フックベクタ数（27行）というラベルでフックするベクタ数を指定して，その数だけコール番号をフックベクタテーブル（919行）に新しい処理アドレスを新ベクタルーチンテーブル（923行）に羅列します。
終了処理（369行〜）では，登録してある垂直同期割り込みの解除とフックベクタを元に戻す処理をしています（398〜403行)。
終了処理に入る前にひとつ注意しなければならないことがあります。コールベクタをフックするプログラム特有のことなのですが，自分でフックしたコールベクタがさらにほかのプログラムによって再フックされている場合，終了処理を行ってはいけな

表1　並べ方その1でのコマのようす

はじめに表示されるコマ	表示されるコマの順番
1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 〜
2	2, 3, 4, 5, 6, 7, 8, 1, 10, 11, 12, 13, 14, 15, 16, 9, 〜
3	3, 4, 5, 6, 7, 8, 1, 2, 11, 12, 13, 14, 15, 16, 9, 10, 〜
4	4, 5, 6, 7, 8, 1, 2, 3, 12, 13, 14, 15, 16, 9, 10, 11, 〜

図2　壁紙動画の並べ方

図3　拡大図

青いボールが飛びまわる

いという決まりがあります。この再フックされているかをチェックしているのがフックベクタ比較ルーチン（709行から）です。

垂直同期割り込みルーチン（887行から）と，新a42e（861行から）が実際の処理内容です。

垂直同期割り込みルーチンでは，一定周期ごとにグラフィックのホーム座標をテキストのホーム座標からの相対座標で設定しています。そして，そのテキストとグラフィックのホーム座標の違いを保つために，新a42eではいったんフック前のルーチンを呼び出したあと，新たにグラフィックのホーム座標を設定し直しています。

さらにひと工夫，壁動玉々.R

テキストの背景を透明にするところまで同じで，そのあと壁動玉々.R（へきどうたまたまと読む）を起動してみてください。

原理は同じです。ハードウェアスクロールしかしてません。まあ，ハードウェアスクロールのみでも，工夫次第でこんなこともできるという例です。

それと，スクロール値がランダムですので立ち上げるたびに違ったアニメーションをします（計算上65536種類あるはずです）。

プログラムの中身は，かなり姑息な処理をしていますので解説は省略します。

チャイルドで壁紙動画.Rを起動したあと，さらに壁紙動画.Rのチャイルドで自分自身を呼び出させる形で常駐します。

●実行ファイルの作り方

作り方は壁紙動画.R，壁動玉々.Rとも同じで，

HAS ～.S
HLK ～.O
CV ～.X

で，作ることができます。

●壁紙動画.R用データの作り方

作り方といっても，私は大部分を手作業で行っているので，たいしたことはしていません。

まず，DōGA PICなどのアニメーションデータを用意し，それを1枚ずつMATIER付属のZOOM.Xで64×64に縮小してPIC.Rでセーブします（ちょっとしたバッチファイルを組むといいかもしれません）。

それが64枚できたら，MATIERを立ち上げ，まずは図1のようにコマデータをすべて表示させます。その後，矩形スクロールを使って図2のように8ドットずつずらしたらできあがりです。

最後に

さてどうでしたでしょうか，究極の無駄遣いといえるでしょうか？　まあ，その辺の判断はおまかせするとします。

本来ならばもっと有効に活用されるべきグラフィックをこのような形でしか活用できないというのは，少し悲しい気もしますが，そこはまあ「細かいことは考えず，面白けりゃなんでもいいじゃん」てな感じでどんどん無駄遣いしてしまいましょう。

リスト1　壁紙動画.S

```
  1: STR@
  2: **********************************************
  3: *    グラフィックで壁紙を動かそう!
  4: *
  5: *              壁紙動画.r  version 1.00
  6: *
  7: *                          by Sho-Ta
  8: *
  9: **********************************************
 10: *          外部ファイル参照
 11:
 12:            .include        doscall.mac
 13:            .include        iocscall.mac
 14:            .include        SXKIT¥SXCALL.MAC
 15:            .include        SXKIT¥SXCALL.H
 16:
 17: **********************************************
 18: *          プログラムエリア
 19:            .text
 20:
 21: **********************************************
 22: *          定数定義
 23: SX_BASIC_SEND   equ     258
 24:
 25: **********************************************
 26: *          ワーク定義
 27: フックベクタ数          equ     1
 28: スタックサイズ          equ     4096
 29:            .offset 0
 30: コードを共有するタスクのID:             .ds.w   1
 31:                                                 .ds.w   1
 32: コマンドラインのアドレス:                .ds.l   1
 33: 環境のアドレス:                                .ds.l   1
 34: 初期化ヘッダ:
 35: イベントレコード:                              .ds.b   TsEvnt
 36: イベントマスク:                                .ds.w   1
 37: スイッチRの値:                                 .ds.w   1
 38: 動画ホーム:                                     .ds.b   Point
 39: 動画スクロール:                                .ds.b   Point
 40: ウェイトカウンタ:                              .ds.w   1
 41: ウェイト値:                                     .ds.w   1
 42: ストップフラグ:                                .ds.w   1
 43: スイッチDフラグ:                               .ds.w   1
 44: 描画ファイル名:                                .ds.b   96
 45: 描画コマンドライン:                           .ds.b   256
 46: 起動フラグ:                                     .ds.w   1
 47: ベクタ退避:                                     .ds.l   フックベクタ数
 48: 初期化テイル:
 49:                                                 .ds.b   スタックサイズ
 50: ワークエリアサイズ:
 51:            .text
 52: ***************************************
 53: モジュールヘッダ:
 54: モジュールタイプ        equ     'OBJR'
 55: モジュールサイズ        equ     モジュールテイル-モジュールヘッダ
 56: スタートオフセット      equ     プログラムスタート-モジュールヘッダ
 57: コモンエリアサイズ      equ     0
 58:            .dc.l   モジュールタイプ
 59:            .dc.l   モジュールサイズ
 60:            .dc.l   スタートオフセット
 61:            .dc.l   ワークエリアサイズ
 62:            .dc.l   コモンエリアサイズ
 63:            .dc.l   0,0,0
 64:
 65: ****************************************
 66: コマンドラインスタート:
 67:            DOS     _EXIT
 68:
 69: ****************************************
 70: プログラムスタート:
 71:            movea.l a1,a5
 72:            move.l  a2,コマンドラインのアドレス(a5)
 73:            move.l  a3,環境のアドレス(a5)
 74:            move.w  d1,コードを共有するタスクのID(a5)
 75:            tst.w   d0
 76:            bmi     起動失敗による終了処理
 77:            bsr     初期化ルーチン
 78:            bmi     起動失敗による終了処理
 79: メインループ:
 80:            pea     イベントレコード(a5)
 81:            move.w  イベントマスク(a5),-(sp)
 82:            SX      __TSEventAvail
 83:            addq.l  #6,sp
 84:            bmi     イベント取得エラー
 85:            lea     イベントジャンプテーブル(pc),a1
 86:            move.w  イベントレコード+tnWhat(a5),d0
 87:            and.w   #$000f,d0
 88:            add.w   d0,d0
 89:            move.w  $00(a1,d0.w),d0
 90:            jsr     $00(a1,d0.w)
 91:            bra     メインループ
 92: イベント取得エラー:
 93:            lea     ワークエリアサイズ(a5),sp
 94:            bra     メインループ
 95: イベントジャンプテーブル:
 96:            .dc.w   アイドルイベントルーチン-イベントジャンプテーブル
 97:            .dc.w   レフトダウンイベントルーチン-イベントジャンプテーブル
 98:            .dc.w   レフトアップイベントルーチン-イベントジャンプテーブル
 99:            .dc.w   ライトダウンイベントルーチン-イベントジャンプテーブル
100:            .dc.w   ライトアップイベントルーチン-イベントジャンプテーブル
101:            .dc.w   キーダウンイベントルーチン-イベントジャンプテーブル
102:            .dc.w   キーアップイベントルーチン-イベントジャンプテーブル
103:            .dc.w   アップデートイベントルーチン-イベントジャンプテーブル
104:            .dc.w   ダミーイベントルーチン-イベントジャンプテーブル
105:            .dc.w   アクティベートイベントルーチン-イベントジャンプテーブル
106:            .dc.w   ダミーイベントルーチン-イベントジャンプテーブル
107:            .dc.w   ダミーイベントルーチン-イベントジャンプテーブル
108:            .dc.w   システム1イベントルーチン-イベントジャンプテーブル
109:            .dc.w   システム2イベントルーチン-イベントジャンプテーブル
110:            .dc.w   ダミーイベントルーチン-イベントジャンプテーブル
111:            .dc.w   ダミーイベントルーチン-イベントジャンプテーブル
112:
113: ****************************************
114: アイドルイベントルーチン:
115: レフトダウンイベントルーチン:
116: レフトアップイベントルーチン:
117: ライトダウンイベントルーチン:
118: ライトアップイベントルーチン:
119: キーダウンイベントルーチン:
120: キーアップイベントルーチン:
121: アップデートイベントルーチン:
122: アクティベートイベントルーチン:
123: ダミーイベントルーチン:
124:            rts
```

▶EYECOMを買いに本屋へ行ったのですが，思わずOh!Xを買ってしまいました。
上山　将史(20)東京都

```
125:
126: ******************************************
127: システム1イベントルーチン:
128: システム2イベントルーチン:
129:         move.w  イベントレコード+tnWhat2(a5),d0
130:         cmp.w   #ENDTSK,d0
131:         beq     システムイベントENDTSK
132:         cmp.w   #SAVE,d0
133:         beq     システムイベントSAVE
134:         cmp.w   #SX_BASIC_SEND,d0
135:         beq     SXBASICイベント
136:         bra     システムイベント終了
137: システムイベントENDTSK:
138: システムイベントENDTSK01:
139:         bsr     フックベクタ比較ルーチン
140:         beq     イベントによる終了処理
141:         cmp.l   #$00000080,イベントレコード+tnWhom2(a5)
142:         bne     システムイベントENDTSK02
143:         pea     終了できないメッセージ(pc)
144:         move.w  #1,-(sp)
145:         SX      __TSErrDialogN
146:         addq.l  #6,sp
147: システムイベントENDTSK02:
148:         bra     システムイベント終了
149: システムイベントENDTSK03:
150:         move.l  #$00000080,イベントレコード+tnWhom2(a5)
151:         bra     システムイベントENDTSK01
152: 終了できないメッセージ:
153:         .dc.b   'ベクタフックが重複していて終了できません。',0
154:         .even
155: システムイベントSAVE:
156:         bsr     コマンドライン設定ルーチン
157:         bra     システムイベント終了
158: システムイベント終了:
159:         rts
160:
161: ******************************************
162: SXBASICイベント:
163:         movea.l イベントレコード+tnWhom(a5),a1
164:         movea.l (a1),a1
165:         lea     メッセージQUIT(pc),a0
166:         bsr     文字列比較ルーチン
167:         beq     システムイベントENDTSK03
168:         lea     メッセージEND(pc),a0
169:         bsr     文字列比較ルーチン
170:         beq     システムイベントENDTSK03
171:         lea     メッセージWAIT(pc),a0
172:         bsr     文字列比較ルーチン
173:         beq     SXBASIC_WAITイベント
174:         lea     メッセージHOME(pc),a0
175:         bsr     文字列比較ルーチン
176:         beq     SXBASIC_HOMEイベント
177:         lea     メッセージSCROLL(pc),a0
178:         bsr     文字列比較ルーチン
179:         beq     SXBASIC_SCROLLイベント
180:         lea     メッセージSTOP(pc),a0
181:         bsr     文字列比較ルーチン
182:         beq     SXBASIC_STOPイベント
183:         lea     メッセージMOVE(pc),a0
184:         bsr     文字列比較ルーチン
185:         beq     SXBASIC_MOVEイベント
186:         lea     メッセージVERSION(pc),a0
187:         bsr     文字列比較ルーチン
188:         beq     SXBASIC_VERSIONイベント
189:         lea     メッセージSTARTUP(pc),a0
190:         bsr     文字列比較ルーチン
191:         beq     SXBASIC_STARTUPイベント
192:         bra     システムイベント終了
193: メッセージQUIT:
194:         .dc.b   'QUIT',0
195: メッセージEND:
196:         .dc.b   'END',0
197: メッセージWAIT:
198:         .dc.b   'WAIT',0
199: メッセージHOME:
200:         .dc.b   'HOME',0
201: メッセージSCROLL:
202:         .dc.b   'SCROLL',0
203: メッセージSTOP:
204:         .dc.b   'STOP',0
205: メッセージMOVE:
206:         .dc.b   'MOVE',0
207: メッセージVERSION:
208:         .dc.b   'VERSION',0
209: メッセージSTARTUP:
210:         .dc.b   'STARTUP',0
211:         .even
212: SXBASIC_WAITイベント:
213:         tst.b   (a1)
214:         beq     SXBASIC_WAITイベント終了
215:         pea     (a1)
216:         SX      $a3df
217:         addq.l  #4,sp
218:         not.b   d0
219:         move.b  d0,ウェイト値+1(a5)
220: SXBASIC_WAITイベント終了:
221:         bra     システムイベント終了
222: SXBASIC_HOMEイベント:
223:         tst.b   (a1)
224:         beq     SXBASIC_HOMEイベント終了
225:         pea     (a1)
226:         SX      $a3df
227:         addq.l  #4,sp
228:         move.w  d0,動画ホーム+ptX(a5)
229:         tst.b   (a0)
230:         beq     SXBASIC_HOMEイベント終了
231:         pea     1(a0)
232:         SX      $a3df
233:         addq.l  #4,sp
234:         move.w  d0,動画ホーム+ptY(a5)
235: SXBASIC_HOMEイベント終了:
236:         clr.l   -(sp)
237:         DOS     _SUPER
238:         move.l  d0,(sp)
239:         bsr     動画ホームセット
240:         DOS     _SUPER
241:         addq.l  #4,sp
242:         bra     システムイベント終了
243: SXBASIC_SCROLLイベント:
244:         tst.b   (a1)
245:         beq     SXBASIC_SCROLLイベント終了
246:         pea     (a1)
247:         SX      $a3df
248:         addq.l  #4,sp
249:         move.w  d0,動画スクロール+ptX(a5)
250:         tst.b   (a0)
251:         beq     SXBASIC_SCROLLイベント終了
252:         pea     1(a0)
253:         SX      $a3df
254:         addq.l  #4,sp
255:         move.w  d0,動画スクロール+ptY(a5)
256: SXBASIC_SCROLLイベント終了:
257:         bra     システムイベント終了
258: SXBASIC_STOPイベント:
259:         move.w  #1,ストップフラグ(a5)
260:         bra     システムイベント終了
261: SXBASIC_MOVEイベント:
262:         clr.w   ストップフラグ(a5)
263:         bra     システムイベント終了
264: SXBASIC_VERSIONイベント:
265:         bsr     通信相手のＩＤ取得ルーチン
266:         clr.l   -(sp)
267:         move.w  #1,-(sp)
268:         move.w  d0,-(sp)
269:         pea     SXBASIC_VERSION送信メッセージ(pc)
270:         movea.l sp,a0
271:         bsr     SXBASICメッセージ送信ルーチン
272:         lea     12(sp),sp
273:         bra     システムイベント終了
274: SXBASIC_VERSION送信メッセージ:
275:         .dc.b   '壁紙動画 VER.1.00',0
276:         .even
277: SXBASIC_STARTUPイベント:
278:         tst.b   (a1)
279:         beq     SXBASIC_STARTUPイベント終了
280:         link    a6,#-task
281:         move.w  #TS_OWN,-(sp)
282:         pea     -task(a6)
283:         SX      __TSGetTdb
284:         addq.l  #6,sp
285:         pea     (a1)
286:         SX      $a3df
287:         addq.l  #4,sp
288:         tst.l   d0
289:         bpl     SXBASIC_STARTUPイベント01
290:         move.b  -task+$1d5(a6),d0
291:         eor.b   #1,d0
292: SXBASIC_STARTUPイベント01:
293:         and.b   #1,d0
294:         and.b   #$fe,-task+$1d5(a6)
295:         or.b    d0,-task+$1d5(a6)
296:         move.w  #TS_OWN,-(sp)
297:         pea     -task(a6)
298:         SX      __TSSetTdb
299:         addq.l  #6,sp
300:         unlk    a6
301: SXBASIC_STARTUPイベント終了:
302:         bra     システムイベント終了
303:
304: ******************************************
305: 初期化ルーチン:
306: 保存レジスタ    reg     d1-d7/a1-a5
307:         movem.l 保存レジスタ,-(sp)
308:         lea     初期化ヘッダ(a5),a0
309:         moveq   #$00,d0
310:         move.w  #(初期化テイル-初期化ヘッダ)/2-1,d1
311: ワーククリアループ:
312:         move.w  d0,(a0)+
313:         dbra    d1,ワーククリアループ
314:         SX      __SXVer
315:         cmp.w   #$010b,d0
316:         bcs     初期化エラー
317:         move.l  a0,d0
318:         cmp.w   #$010b,d0
319:         bcs     初期化エラー
320:         move.w  #$f800,イベントマスク(a5)
321:         move.l  #$0010_0080,動画スクロール(a5)
322:         move.w  #$00ff,ウェイトカウンタ(a5)
323:         move.w  #$0001,-(sp)
324:         clr.l   -(sp)
325:         clr.l   -(sp)
326:         move.l  コマンドラインのアドレス(a5),-(sp)
327:         SX      __TSTakeParam
328:         lea     14(sp),sp
329:         bmi     初期化終了
330:         movea.l a0,a2
331:         pea     (a2)
332:         movea.l sp,a0
333:         bsr     コマンドライン解析ルーチン
334:         SX      __MMDisposePtr
335:         addq.l  #4,sp
336:         tst.w   スイッチRの値(a5)
337:         bne     スイッチR指定による終了処理
338:         tst.w   コードを共有するタスクのＩＤ(a5)
339:         bne     二重起動による終了処理
340:         lea     ワークエリアポインタ(pc),a0
341:         move.l  a5,(a0)
342:         move.w  #$0001,起動フラグ(a5)
343:         lea     フックベクタテーブル(pc),a1
344:         lea     新ベクタルーチンテーブル(pc),a2
345:         move.l  a2,d1
346:         lea     ベクタ退避(a5),a3
347:         moveq   #フックベクタ数,d3
348: ベクタフックループ:
349:         move.l  (a2)+,-(sp)
350:         add.l   d1,(sp)
351:         move.w  (a1)+,-(sp)
352:         SX      __SXSetVector
353:         addq.l  #6,sp
354:         move.l  d0,(a3)+
355:         subq.l  #1,d3
356:         bne     ベクタフックループ
357:         pea     垂直同期割り込みルーチン(pc)
358:         SX      __EXEnVDISPST
359:         addq.l  #4,sp
360:         moveq   #$00,d0
361: 初期化終了:
362:         movem.l (sp)+,保存レジスタ
363:         rts
364: 初期化エラー:
365:         moveq   #$ff,d0
366:         bra     初期化終了
367:
368: ******************************************
369: スイッチR指定による終了処理:
370:         moveq   #0,d2
371:         move.w  コードを共有するタスクのＩＤ(a5),d0
372:         beq     終了処理終了
373:         move.w  d0,-(sp)
374:         clr.w   -(sp)
375:         move.w  #$0001,-(sp)
376:         move.l  #$00000080,-(sp)
377:         clr.l   -(sp)
378:         SX      __TSPostEventTsk2
379:         lea     14(sp),sp
380: 二重起動による終了処理:
381:         moveq   #$00,d2
382:         bra     終了処理終了
383: イベントによる終了処理:
384:         moveq   #$00,d0
385: 起動失敗による終了処理:
386:         move.l  d0,d2
```

▶久しぶりにOh!Xを買った。プログラムのテクニックで勝負する(できる)パソコンは，もうX68000だけでしょう。他機種はマシンスペックに頼ってますから……。

小池　靖(24)愛媛県

```
387:            tst.w   起動フラグ(a5)
388:            beq     終了処理終了
389:            pea     垂直同期割り込みルーチン(pc)
390:            SX      __EXDeVDISPST
391:            addq.l  #4,sp
392:            lea     ワークエリアポインタ(pc),a0
393:            clr.l   (a0)
394:            lea     フックベクタテーブル(pc),a1
395:            lea     ベクタ退避(a5),a2
396:            moveq   #フックベクタ数,d2
397: フックベクタ復帰ループ:
398:            move.l  (a2)+,-(sp)
399:            move.w  (a1)+,-(sp)
400:            SX      __SXSetVector
401:            addq.l  #6,sp
402:            subq.l  #1,d2
403:            bne     フックベクタ復帰ループ
404:            SX      $a42f
405:            move.l  d0,-(sp)
406:            SX      $a42e
407:            addq.l  #4,sp
408: 終了処理終了:
409:            move.l  d2,-(sp)
410:            SX      __TSExit
411:
412: ******************************************
413: コマンドライン解析ルーチン:
414: 保存レジスタ     reg      d1-d7/a1-a5
415:            movem.l 保存レジスタ,-(sp)
416:            movea.l a0,a4
417:            movea.l $0000(a4),a2
418:            move.l  (a2)+,d2
419: コマンドライン解析01:
420:            subq.l  #1,d2
421:            bmi     コマンドライン解析終了
422:            movea.l (a2)+,a1
423:            move.b  (a1),d0
424:            beq     コマンドライン解析01
425:            cmpi.b  #'-',d0
426:            beq     コマンドライン解析02
427:            cmpi.b  #'/',d0
428:            bne     コマンドライン解析ff
429: コマンドライン解析02:
430:            addq.l  #1,a1
431:            move.b  (a1)+,d0
432:            and.b   #$5f,d0
433:            cmpi.b  #'R',d0
434:            bne     コマンドライン解析03
435:            move.w  #$0001,スイッチRの値(a5)
436:            bra     コマンドライン解析ff
437: コマンドライン解析03:
438:            cmpi.b  #'M',d0
439:            bne     コマンドライン解析04
440:            move.l  a5,-(sp)
441:            tst.w   コードを共有するタスクのID(a5)
442:            beq     コマンドライン解析03_01
443:            move.l  ワークエリアポインタ(pc),d0
444:            beq     コマンドライン解析03_01
445:            movea.l d0,a5
446: コマンドライン解析03_01:
447:            pea     (a1)
448:            SX      $a3df
449:            addq.l  #4,sp
450:            tst.l   d0
451:            bpl     コマンドライン解析03_02
452:            move.w  ストップフラグ(a5),d0
453:            eor.w   #1,d0
454: コマンドライン解析03_02:
455:            and.w   #1,d0
456:            move.w  d0,ストップフラグ(a5)
457:            move.l  (sp)+,a5
458:            bra     コマンドライン解析ff
459: コマンドライン解析04:
460:            cmpi.b  #'H',d0
461:            bne     コマンドライン解析05
462:            move.l  a5,-(sp)
463:            tst.w   コードを共有するタスクのID(a5)
464:            beq     コマンドライン解析04_01
465:            move.l  ワークエリアポインタ(pc),d0
466:            beq     コマンドライン解析04_01
467:            movea.l d0,a5
468: コマンドライン解析04_01:
469:            pea     (a1)
470:            SX      $a3df
471:            addq.l  #4,sp
472:            move.w  d0,動画ホーム+ptX(a5)
473:            pea     1(a0)
474:            SX      $a3df
475:            addq.l  #4,sp
476:            move.w  d0,動画ホーム+ptY(a5)
477:            move.l  (sp)+,a5
478:            bra     コマンドライン解析ff
479: コマンドライン解析05:
480:            cmpi.b  #'S',d0
481:            bne     コマンドライン解析06
482:            move.l  a5,-(sp)
483:            tst.w   コードを共有するタスクのID(a5)
484:            beq     コマンドライン解析05_01
485:            move.l  ワークエリアポインタ(pc),d0
486:            beq     コマンドライン解析05_01
487:            movea.l d0,a5
488: コマンドライン解析05_01:
489:            pea     (a1)
490:            SX      $a3df
491:            addq.l  #4,sp
492:            move.w  d0,動画スクロール+ptX(a5)
493:            pea     1(a0)
494:            SX      $a3df
495:            addq.l  #4,sp
496:            move.w  d0,動画スクロール+ptY(a5)
497:            move.l  (sp)+,a5
498:            bra     コマンドライン解析ff
499: コマンドライン解析06:
500:            cmpi.b  #'A',d0
501:            bne     コマンドライン解析07
502:            move.l  a5,-(sp)
503:            tst.w   コードを共有するタスクのID(a5)
504:            beq     コマンドライン解析06_01
505:            move.l  ワークエリアポインタ(pc),d0
506:            beq     コマンドライン解析06_01
507:            movea.l d0,a5
508: コマンドライン解析06_01:
509:            pea     (a1)
510:            SX      $a3df
511:            addq.l  #4,sp
512:            not.b   d0
513:            move.b  d0,ウェイト値+1(a5)
514:            move.l  (sp)+,a5
515:            bra     コマンドライン解析ff
516: コマンドライン解析07:
517: コマンドライン解析08:
518:            cmpi.b  #'K',d0
519:            bne     コマンドライン解析09
520:            link    a6,#-2-task
521:            move.w  コードを共有するタスクのID(a5),d0
522:            bne     コマンドライン解析08_01
523:            move.w  #TS_OWN,d0
524: コマンドライン解析08_01:
525:            move.w  d0,-2-task(a6)
526:            move.w  -2-task(a6),-(sp)
527:            pea     -task(a6)
528:            SX      __TSGetTdb
529:            addq.l  #6,sp
530:            pea     (a1)
531:            SX      $a3df
532:            addq.l  #4,sp
533:            tst.l   d0
534:            bpl     コマンドライン解析08_02
535:            move.b  -task+$1d5(a6),d0
536:            eor.b   #1,d0
537: コマンドライン解析08_02:
538:            and.b   #1,d0
539:            and.b   #$fe,-task+$1d5(a6)
540:            or.b    d0,-task+$1d5(a6)
541:            move.w  -2-task(a6),-(sp)
542:            pea     -task(a6)
543:            SX      __TSSetTdb
544:            addq.l  #6,sp
545:            unlk    a6
546:            bra     コマンドライン解析ff
547: コマンドライン解析09:
548:            cmpi.b  #'D',d0
549:            bne     コマンドライン解析0a
550:            move.l  a5,-(sp)
551:            tst.w   コードを共有するタスクのID(a5)
552:            beq     コマンドライン解析09_01
553:            move.l  ワークエリアポインタ(pc),d0
554:            beq     コマンドライン解析09_01
555:            movea.l d0,a5
556: コマンドライン解析09_01:
557:            move.w  #1,スイッチDフラグ(a5)
558:            pea     描画ファイル名(a5)
559:            pea     (a1)
560:            SX      __SXStrCopy
561:            addq.l  #8,sp
562:            lea     描画コマンドライン+1(a5),a0
563:            move.l  a0,d1
564:            clr.b   (a0)
565: コマンドライン解析09_02:
566:            subq.l  #1,d2
567:            bmi     コマンドライン解析09_03
568:            move.b  #' ',(a0)+
569:            pea     (a0)
570:            move.l  (a2)+,-(sp)
571:            SX      __SXStrCopy
572:            addq.l  #8,sp
573:            bra     コマンドライン解析09_02
574: コマンドライン解析09_03:
575:            move.l  a0,d0
576:            sub.b   d1,d0
577:            move.b  d0,描画コマンドライン(a5)
578:            pea     描画ファイル名(a5)
579:            clr.l   -(sp)
580:            pea     描画コマンドライン(a5)
581:            pea     描画ファイル名(a5)
582:            move.w  #0<<8+0,-(sp)
583:            SX      __TSFockB
584:            lea     18(sp),sp
585:            tst.l   d0
586:            bmi     コマンドライン解析09_04
587:            link    a6,#-2-task-TsEvnt
588:            move.w  d0,-2-task-TsEvnt(a6)
589:            move.w  -2-task-TsEvnt(a6),-(sp)
590:            pea     -task-TsEvnt(a6)
591:            SX      __TSGetTdb
592:            addq.l  #6,sp
593:            bset    #0,-task+$1d5-TsEvnt(a6)
594:            move.w  -2-task-TsEvnt(a6),-(sp)
595:            pea     -task-TsEvnt(a6)
596:            SX      __TSSetTdb
597:            addq.l  #6,sp
598:            clr.l   -TsEvnt+tnWhom(a6)
599:            clr.l   -TsEvnt+tnWhom2(a6)
600:            move.w  #STARTUP,-TsEvnt+tnWhat2(a6)
601:            pea     -TsEvnt(a6)
602:            move.w  -2-task-TsEvnt(a6),-(sp)
603:            SX      __TSSendMes
604:            addq.l  #6,sp
605:            unlk    a6
606: コマンドライン解析09_04:
607:            move.l  (sp)+,a5
608:            bra     コマンドライン解析終了
609: コマンドライン解析0a:
610: コマンドライン解析ff:
611:            bra     コマンドライン解析01
612: コマンドライン解析終了:
613:            movem.l (sp)+,保存レジスタ
614:            rts
615:
616: ******************************************
617: コマンドライン設定ルーチン:
618: 保存レジスタ     reg      d2/a0
619:            link    a6,#-task
620:            move.w  #TS_OWN,-(sp)
621:            pea     -task(a6)
622:            SX      __TSGetTdb
623:            addq.l  #6,sp
624:            lea     -task+tsCommand+1(a6),a0
625:            move.l  a0,d2
626:            move.b  #'-',(a0)+
627:            move.b  #'M',(a0)+
628:            move.w  ストップフラグ(a5),d0
629:            ext.l   d0
630:            move.l  d0,-(sp)
631:            pea     (a0)
632:            SX      $a3e0
633:            addq.l  #8,sp
634:            move.b  #' ',(a0)+
635:            move.b  #'-',(a0)+
636:            move.b  #'H',(a0)+
637:            move.w  動画ホーム+ptX(a5),d0
638:            and.l   #1023,d0
639:            move.l  d0,-(sp)
640:            pea     (a0)
641:            SX      $a3e0
642:            addq.l  #8,sp
643:            move.b  #',',(a0)+
644:            move.w  動画ホーム+ptY(a5),d0
645:            and.l   #1023,d0
646:            move.l  d0,-(sp)
647:            pea     (a0)
648:            SX      $a3e0
```

▶ジャストの広告再開。石上氏のアクセラレータも動いたようで，今後が楽しみです。
冨田　英明(19)愛知県

```
649:          addq.l  #8,sp
650:          move.b  #' ',(a0)+
651:          move.b  #'-',(a0)+
652:          move.b  #'S',(a0)+
653:          move.w  動画スクロール+ptX(a5),d0
654:          ext.l   d0
655:          move.l  d0,-(sp)
656:          pea     (a0)
657:          SX      $a3e0
658:          addq.l  #8,sp
659:          move.b  #',',(a0)+
660:          move.w  動画スクロール+ptY(a5),d0
661:          ext.l   d0
662:          move.l  d0,-(sp)
663:          pea     (a0)
664:          SX      $a3e0
665:          addq.l  #8,sp
666:          move.b  #' ',(a0)+
667:          move.b  #'-',(a0)+
668:          move.b  #'A',(a0)+
669:          move.w  ウェイト値(a5),d0
670:          not.b   d0
671:          ext.l   d0
672:          move.l  d0,-(sp)
673:          pea     (a0)
674:          SX      $a3e0
675:          addq.l  #8,sp
676:          move.b  #' ',(a0)+
677:          move.b  #'-',(a0)+
678:          move.b  #'K',(a0)+
679:          clr.l   -(sp)
680:          pea     (a0)
681:          SX      $a3e0
682:          addq.l  #8,sp
683:          tst.w   スイッチDフラグ(a5)
684:          beq     描画コマンド無し
685:          move.b  #' ',(a0)+
686:          move.b  #'-',(a0)+
687:          move.b  #'D',(a0)+
688:          pea     (a0)
689:          pea     描画ファイル名(a5)
690:          SX      __SXStrCopy
691:          addq.l  #8,sp
692:          pea     (a0)
693:          pea     描画コマンドライン+1(a5)
694:          SX      __SXStrCopy
695:          addq.l  #8,sp
696: 描画コマンド無し:
697:          move.l  a0,d0
698:          sub.b   d2,d0
699:          move.b  d0,-task+tsCommand(a6)
700:          clr.b   (a0)
701:          move.w  #TS_OWN,-(sp)
702:          pea     -task(a6)
703:          SX      __TSSetTdb
704:          addq.l  #6,sp
705:          unlk    a6
706:          rts
707:
708: ******************************************
709: フックベクタ比較ルーチン:
710: 保存レジスタ    reg     d1-d7/a1-a5
711:          movem.l 保存レジスタ,-(sp)
712:          lea     フックベクタテーブル(pc),a1
713:          lea     新ベクタルーチンテーブル(pc),a2
714:          move.l  a2,d1
715:          moveq   #フックベクタ数,d2
716: フックベクタ比較ループ:
717:          move.w  (a1)+,-(sp)
718:          SX      __SXGetVector
719:          addq.l  #2,sp
720:          sub.l   d1,d0
721:          cmp.l   (a2)+,d0
722:          bne     フックベクタ比較エラー
723:          subq.l  #1,d2
724:          bne     フックベクタ比較ループ
725:          moveq   #$00,d0
726: フックベクタ比較終了:
727:          movem.l (sp)+,保存レジスタ
728:          rts
729: フックベクタ比較エラー:
730:          moveq   #$ff,d0
731:          bra     フックベクタ比較終了
732:
733: ******************************************
734: 通信相手のID取得ルーチン:
735:          link    a6,#-task
736:          move.l  a0,-(sp)
737:          move.w  #TS_OWN,-(sp)
738:          pea     -task(a6)
739:          SX      __TSGetTdb
740:          addq.l  #6,sp
741:          move.w  -task+tsCommRecvID(a6),d0
742:          movea.l (sp)+,a0
743:          unlk    a6
744:          rts
745:
746: ******************************************
747: SXBASICメッセージ送信ルーチン:
748: *        (a0).l  = メッセージポインタ
749: *        4(a0).w = 転送先タスクID
750: *        6(a0).w = TSAnswer 実行フラグ
751: *        8(a0).w = 返事受け取り用イベントレコードポインタ
752: 保存レジスタ    reg     d1/a0/a4
753:          link    a6,#-4-TsEvnt-4-TsEvnt
754:          movem.l 保存レジスタ,-(sp)
755:          movea.l a0,a4
756:          movea.l (a4),a0
757:          move.l  a0,d1
758: SXBASICメッセージ長さ調べ:
759:          tst.b   (a0)+
760:          bne     SXBASICメッセージ長さ調べ
761:          move.l  a0,d0
762:          sub.l   d1,d0
763:          move.l  d0,-4-TsEvnt-4-TsEvnt(a6)
764:          move.l  (a4),a0
765:          move.l  a0,-4-TsEvnt(a6)
766:          lea     -4-TsEvnt(a6),a0
767:          move.l  a0,-TsEvnt+tnWhom(a6)
768:          move.w  #SX_BASIC_SEND,-TsEvnt+tnWhat2(a6)
769:          tst.w   6(a4)
770:          beq     SXBASICメッセージ送信01
771:          clr.l   -(sp)
772:          move.l  -4-TsEvnt-4-TsEvnt(a6),-(sp)
773:          move.w  4(a4),-(sp)
774:          move.w  #3<<8+0,-(sp)
775:          move.w  -TsEvnt+tnWhat2(a6),-(sp)
776:          move.l  -TsEvnt+tnWhom2(a6),-(sp)
777:          move.l  -TsEvnt+tnWhom(a6),-(sp)
778:          SX      __TSAnswer2
779:          lea     22(sp),sp
```

```
780:          clr.l   -(sp)
781:          move.l  -4-TsEvnt-4-TsEvnt(a6),-(sp)
782:          move.w  4(a4),-(sp)
783:          move.w  #3<<8+0,-(sp)
784:          move.w  -TsEvnt+tnWhat2(a6),-(sp)
785:          move.l  -TsEvnt+tnWhom2(a6),-(sp)
786:          move.l  -TsEvnt+tnWhom(a6),-(sp)
787:          SX      __TSPostEventTsk3
788:          lea     22(sp),sp
789:          bra     SXBASICメッセージ送信成功
790: SXBASICメッセージ送信01:
791:          moveq   #10-1,d1
792: SXBASICメッセージ送信ループ:
793:          pea     -TsEvnt(a6)
794:          move.w  4(a4),-(sp)
795:          SX      __TSSendMes
796:          addq.l  #6,sp
797:          tst.l   d0
798:          bpl     SXBASICメッセージ送信成功
799:          dbra    d1,SXBASICメッセージ送信ループ
800: SXBASICメッセージ送信成功:
801:          move.l  d0,d1
802:          cmp.l   #2,d1
803:          bne     SXBASICメッセージ送信02
804:          move.l  8(a4),d0
805:          beq     SXBASICメッセージ送信02
806:          movea.l d0,a0
807:          move.w  -TsEvnt+tnWhat(a6),tnWhat(a0)
808:          move.l  -TsEvnt+tnWhom(a6),tnWhom(a0)
809:          move.l  -TsEvnt+tnWhen(a6),tnWhen(a0)
810:          move.l  -TsEvnt+tnWhom2(a6),tnWhom2(a0)
811:          move.l  -TsEvnt+tnWhat2(a6),tnWhat2(a0)
812: SXBASICメッセージ送信02:
813:          move.l  d1,d0
814:          movem.l (sp)+,保存レジスタ
815:          unlk    a6
816:          rts
817:
818: ******************************************
819: 文字列比較ルーチン:
820: *        引数 a0.l = 文字列1のアドレス(ASCIIZ)
821: *             a1.l = 文字列2のアドレス(区切り:0,CR,SPACE,TAB)
822: *        一致した場合 a0.l = 文字列2のアドレス
823: *                     a1.l = 文字列2の次の文字列のアドレス
824:          move.l  a1,-(sp)
825: 文字列比較ループ:
826:          move.b  (a0)+,d0
827:          beq     文字列比較01
828:          cmp.b   (a1)+,d0
829:          bne     文字列比較不一致
830:          bra     文字列比較ループ
831: 文字列比較01:
832:          move.b  (a1)+,d0
833:          beq     文字列比較一致
834:          cmpi.b  #$0d,d0
835:          beq     文字列比較一致
836:          cmpi.b  #$09,d0
837:          beq     文字列比較一致
838:          cmpi.b  #' ',d0
839:          beq     文字列比較一致
840:          cmpi.b  #$1a,d0
841:          beq     文字列比較一致
842: 文字列比較不一致:
843:          moveq   #-1,d0
844:          move.l  (sp)+,a1
845:          rts
846: 文字列比較一致:
847:          move.l  (sp)+,a0
848: 次の文字列の先頭を探す:
849:          move.b  (a1)+,d0
850:          cmpi.b  #$0d,d0
851:          beq     次の文字列の先頭を探す
852:          cmpi.b  #$09,d0
853:          beq     次の文字列の先頭を探す
854:          cmpi.b  #$20,d0
855:          beq     次の文字列の先頭を探す
856:          subq.l  #1,a1
857:          moveq   #0,d0
858:          rts
859:
860: ******************************************
861: 新a42e:
862:          move.l  a0,-(sp)
863:          movea.l ワークエリアポインタ(pc),a0
864:          move.l  ベクタ退避(a0),d0
865:          movea.l (sp)+,a0
866:          pea     動画ホームセット(pc)
867:          move.l  d0,-(sp)
868:          rts
869: 動画ホームセット:
870:          move.l  d0,-(sp)
871:          move.l  a5,-(sp)
872:          movea.l ワークエリアポインタ(pc),a5
873:          SX      $a42f
874:          swap    d0
875:          add.w   動画ホーム+ptX(a5),d0
876:          swap    d0
877:          add.w   動画ホーム+ptY(a5),d0
878:          move.l  d0,$e80018
879:          move.l  d0,$e8001c
880:          move.l  d0,$e80020
881:          move.l  d0,$e80024
882:          movea.l (sp)+,a5
883:          move.l  (sp)+,d0
884:          rts
885:
886: ******************************************
887: 垂直同期割り込みルーチン:
888:          movea.l ワークエリアポインタ(pc),a5
889:          tas.b   起動フラグ(a5)
890:          beq     垂直同期割り込み終了
891:          tst.w   ストップフラグ(a5)
892:          bne     垂直同期割り込み01
893:          move.w  ウェイトカウンタ(a5),d0
894:          sub.w   ウェイト値(a5),d0
895:          move.w  d0,ウェイトカウンタ(a5)
896:          bpl     垂直同期割り込み01
897:          add.w   #$00ff,d0
898:          move.w  d0,ウェイトカウンタ(a5)
899:          move.l  動画ホーム(a5),d0
900:          swap    d0
901:          add.w   動画スクロール+ptX(a5),d0
902:          swap    d0
903:          add.w   動画スクロール+ptY(a5),d0
904:          move.l  d0,動画ホーム(a5)
905: 垂直同期割り込み01:
906:          SX      $a42f
907:          swap    d0
908:          add.w   動画ホーム+ptX(a5),d0
909:          swap    d0
910:          add.w   動画ホーム+ptY(a5),d0
```

▶「スタートレックーネクストジェネレーションー」は面白いです。
隈田　章寛(22)愛知県

```
911:         move.l  d0,$e80018
912:         move.l  d0,$e8001c
913:         move.l  d0,$e80020
914:         move.l  d0,$e80024
915: 垂直同期割り込み終了:
916:         rts
917:
918: ********************************************
919: フックベクタテーブル:
920:         .dc.w   $a42e
921:
922: ********************************************
923: 新ベクタルーチンテーブル:
924:         .dc.l   新a42e-新ベクタルーチンテーブル
925:
926: ********************************************
927: ワークエリアポインタ:
928:         .dc.l   $00000000
929:
930: **************************************************
931: モジュールテイル:
932:
933:         .end    コマンドラインスタート
```

リスト2 壁動玉々.S

```
  1: *(注)シャーペンテキストです
  2: **************************************************
  3: *      壁紙動画.r 用サブプログラム
  4: *               へきどうたまたま
  5: *               壁動玉々.r ver.0.90
  6: *
  7: *                   by Sho-Ta
  8: **************************************************
  9: *       外部ファイル参照
 10:
 11:         .include        doscall.mac
 12:         .include        iocscall.mac
 13:         .include        SXKIT¥SXCALL.MAC
 14:         .include        SXKIT¥SXCALL.H
 15:
 16: **************************************************
 17: *       プログラムエリア
 18:         .text
 19:
 20: **************************************************
 21: *       定数定義
 22: SX_BASIC_SEND   equ     258
 23:
 24: **************************************************
 25: *       ワーク定義
 26: スタックサイズ          equ     4096
 27:         .offset 0
 28: コードを共有するタスクのID:             .ds.w   1
 29:                                                 .ds.w   1
 30: コマンドラインのアドレス:               .ds.l   1
 31: 環境のアドレス:                           .ds.l   1
 32: 初期化ヘッダ:
 33: スイッチGフラグ:                          .ds.w   1
 34: スイッチMフラグ:                          .ds.w   1
 35: スイッチA値:                               .ds.w   1
 36: スイッチS値:                               .ds.b   Point
 37: スイッチKフラグ:                          .ds.w   1
 38: スイッチK値:                               .ds.w   1
 39: 初期化テイル:
 40:                                                 .ds.b   スタックサイズ
 41: ワークエリアサイズ:
 42:         .text
 43: ********************************************
 44: モジュールヘッダ:
 45: モジュールタイプ        equ     'OBJR'
 46: モジュールサイズ        equ     モジュールテイル-モジュールヘッダ
 47: スタートオフセット      equ     プログラムスタート-モジュールヘッダ
 48: コモンエリアサイズ      equ     0
 49:         .dc.l   モジュールタイプ
 50:         .dc.l   モジュールサイズ
 51:         .dc.l   スタートオフセット
 52:         .dc.l   ワークエリアサイズ
 53:         .dc.l   コモンエリアサイズ
 54:         .dc.l   0,0,0
 55:
 56: ********************************************
 57: コマンドラインスタート:
 58:         DOS     _EXIT
 59:
 60: ********************************************
 61: プログラムスタート:
 62:         movea.l a1,a5
 63:         move.l  a2,コマンドラインのアドレス(a5)
 64:         move.l  a3,環境のアドレス(a5)
 65:         move.w  d1,コードを共有するタスクのID(a5)
 66:         tst.w   d0
 67:         bmi     起動失敗による終了処理
 68:         bsr     初期化ルーチン
 69:         bmi     起動失敗による終了処理
 70:         bsr     グラフィックモードチェックルーチン
 71:         bmi     起動失敗による終了処理
 72:         bsr     チャイルドチェックルーチン
 73:         bmi     壁紙動画起動
 74:         bsr     描画ルーチン
 75:         bra     正常終了処理
 76: 壁紙動画起動:
 77:         bsr     壁紙動画起動ルーチン
 78:         bra     正常終了処理
 79:
 80: ********************************************
 81: 壁紙動画起動ルーチン:
 82: 保存レジスタ    reg     d1-d7/a1-a5
 83:         link    a6,#-256-2-TsEvnt-task
 84:         movem.l 保存レジスタ,-(sp)
 85:
 86:         lea     -256+1-2-TsEvnt-task(a6),a0
 87:         move.l  a0,d2
 88:         pea     (a0)
 89:         pea     壁紙動画起動スイッチ文字列1(pc)
 90:         SX      __SXStrCopy
 91:         addq.l  #8,sp
 92:         move.b  #' ',(a0)+
 93:         move.b  #'-',(a0)+
 94:         move.b  #'S',(a0)+
 95:         move.w  スイッチS値+ptX(a5),d0
 96:         and.l   #511,d0
 97:         move.l  d0,-(sp)
 98:         pea     (a0)
 99:         SX      $a3e0
100:         addq.l  #8,sp
101:         move.b  #',',(a0)+
102:         move.w  スイッチS値+ptY(a5),d0
103:         and.l   #511,d0
104:         move.l  d0,-(sp)
105:         pea     (a0)
106:         SX      $a3e0
107:         addq.l  #8,sp
108:         move.b  #' ',(a0)+
109:         move.b  #'-',(a0)+
110:         move.b  #'A',(a0)+
111:         move.w  スイッチA値(a5),d0
112:         and.l   #511,d0
113:         move.l  d0,-(sp)
114:         pea     (a0)
115:         SX      $a3e0
116:         addq.l  #8,sp
117:         move.b  #' ',(a0)+
118:         move.b  #'-',(a0)+
119:         move.b  #'K',(a0)+
120:         move.l  a0,-(sp)
121:         move.w  #TS_OWN,-(sp)
122:         pea     -task(a6)
123:         SX      __TSGetTdb
124:         addq.l  #6,sp
125:         movea.l (sp)+,a0
126:         moveq   #0,d0
127:         move.b  -task+$1d5(a6),d0
128:         tst.w   スイッチKフラグ(a5)
129:         beq     スイッチK無し
130:         move.b  スイッチK値+1(a5),d0
131: スイッチK無し:
132:         and.b   #$01,d0
133:         move.l  d0,-(sp)
134:         pea     (a0)
135:         SX      $a3e0
136:         addq.l  #8,sp
137:         pea     (a0)
138:         pea     壁紙動画起動スイッチ文字列2(pc)
139:         SX      __SXStrCopy
140:         addq.l  #8,sp
141:         move.b  #' ',(a0)+
142:         move.b  #'-',(a0)+
143:         move.b  #'S',(a0)+
144:         move.w  スイッチS値+ptX(a5),d0
145:         and.l   #511,d0
146:         move.l  d0,-(sp)
147:         pea     (a0)
148:         SX      $a3e0
149:         addq.l  #8,sp
150:         move.b  #',',(a0)+
151:         move.w  スイッチS値+ptY(a5),d0
152:         and.l   #511,d0
153:         move.l  d0,-(sp)
154:         pea     (a0)
155:         SX      $a3e0
156:         addq.l  #8,sp
157:         move.l  a0,d0
158:         sub.b   d2,d0
159:         move.b  d0,-256-2-TsEvnt-task(a6)
160:         clr.b   (a0)
161:         pea     名前壁紙動画(pc)
162:         clr.l   -(sp)
163:         pea     -256-2-TsEvnt-task(a6)
164:         pea     名前壁紙動画(pc)
165:         move.w  #0<<8+0,-(sp)
166:         SX      __TSFockB
167:         lea     18(sp),sp
168:         tst.l   d0
169:         bmi     壁紙起動エラー
170:         move.w  d0,-2-TsEvnt-task(a6)
171:         clr.l   -TsEvnt+tnWhom-task(a6)
172:         clr.l   -TsEvnt+tnWhom2-task(a6)
173:         move.w  #STARTUP,-TsEvnt+tnWhat2-task(a6)
174:         pea     -TsEvnt-task(a6)
175:         move.w  -2-TsEvnt-task(a6),-(sp)
176:         SX      __TSSendMes
177:         addq.l  #6,sp
178:         moveq   #0,d0
179: 壁紙起動終了:
180:         movem.l (sp)+,保存レジスタ
181:         unlk    a6
182:         rts
183: 壁紙起動エラー:
184:         tst.w   スイッチMフラグ(a5)
185:         bne     壁紙起動エラー01
186:         pea     壁紙起動エラーメッセージ(pc)
187:         move.w  #1,-(sp)
188:         SX      __TSErrDialogN
189:         addq.l  #6,sp
190: 壁紙起動エラー01:
191:         moveq   #-1,d0
192:         bra     壁紙起動終了
193: 壁紙動画起動スイッチ文字列1:
194:         .dc.b   '-M0 -H0,0',0
195: 壁紙動画起動スイッチ文字列2:
196:         .dc.b   ' -D壁動玉々.r',0
197: 壁紙起動エラーメッセージ:
198:         .dc.b   '壁紙動画.r の起動に失敗しました。',0
199:         .even
200:
201: ********************************************
202: チャイルドチェックルーチン:
203: 保存レジスタ    reg     d1-d7/a1-a5
204:         link    a6,#-task
205:         movem.l 保存レジスタ,-(sp)
206:         move.w  #TS_OWN,-(sp)
207:         pea     -task(a6)
208:         SX      __TSGetTdb
209:         addq.l  #6,sp
210:         moveq   #-1,d0
211: チャイルドチェックルーチン01:
212:         move.w  d0,-(sp)
213:         pea     名前壁紙動画(pc)
214:         SX      __TSFindTskn
215:         addq.l  #6,sp
216:         tst.w   d0
217:         bmi     壁紙動画のチャイルドではない
218:         cmp.w   -task+tsParentID(a6),d0
219:         bne     チャイルドチェックルーチン01
220:         moveq   #0,d0
221: チャイルドチェックルーチン終了:
222:         movem.l (sp)+,保存レジスタ
223:         unlk    a6
224:         rts
225: 壁紙動画のチャイルドではない:
226:         moveq   #-1,d0
```

▶ 7月号の「ショートプロぱーてぃ」に載っていた“TYPE_V30.BAS”は長掌筋(?)が痙って楽しいぞ！　太田　志輝(17)北海道

```
227:         bra     チャイルドチェックルーチン終了
228: 名前壁紙動画:
229:         .dc.b   '壁紙動画.r',0
230:         .even
231:
232: ******************************************
233: グラフィックモードチェックルーチン:
234: 保存レジスタ    reg     d1-d7/a1-a5
235:         movem.l 保存レジスタ,-(sp)
236: グラフィックモードチェック01:
237:         SX      __TSGetGraphMode
238:         move.w  a0,d0
239:         lea     色モードテーブル(pc),a0
240:         move.b  $00(a0,d0.w),d0
241:         cmp.b   #3,d0
242:         beq     グラフィックモードチェック0f
243:         tst.w   スイッチGフラグ(a5)
244:         beq     グラフィックモードチェックエラー
245:         move.w  #-1,-(sp)
246:         pea     名前SAdjust(pc)
247:         SX      __TSFindTskn
248:         addq.l  #6,sp
249:         tst.w   d0
250:         bmi     グラフィックモードチェックエラー
251:         link    a6,#-2-4-TsEvnt
252:         move.w  d0,-2-4 TsEvnt(a6)
253:         lea     SXBASIC_GCOLOR送信メッセージ(pc),a0
254:         move.l  a0,-4-TsEvnt(a6)
255:         lea     -4-TsEvnt(a6),a0
256:         move.l  a0,-TsEvnt+tnWhom(a6)
257:         move.w  #SX_BASIC_SEND,-TsEvnt+tnWhat2(a6)
258:         moveq   #10-1,d1
259: グラフィックモードチェック02:
260:         pea     -TsEvnt(a6)
261:         move.w  -2-4-TsEvnt(a6),-(sp)
262:         SX      __TSSendMes
263:         addq.l  #6,sp
264:         tst.l   d0
265:         bpl     グラフィックモードチェック03
266:         dbra    d1,グラフィックモードチェック02
267: グラフィックモードチェック03:
268:         unlk    a6
269:         clr.w   スイッチGフラグ(a5)
270:         bra     グラフィックモードチェック01
271: グラフィックモードチェック0f:
272:         moveq   #0,d0
273: グラフィックモードチェック終了:
274:         movem.l (sp)+,保存レジスタ
275:         rts
276: グラフィックモードチェックエラー:
277:         tst.w   スイッチMフラグ(a5)
278:         bne     グラフィックモードチェックエラー01
279:         pea     グラフィックモードチェックエラーメッセージ(pc)
280:         move.w  #1,-(sp)
281:         SX      __TSErrDialogN
282:         addq.l  #6,sp
283: グラフィックモードチェックエラー01:
284:         moveq   #-1,d0
285:         bra     グラフィックモードチェック終了
286: 色モードテーブル:
287:         .dc.b   4,4,4,4,0,0,0,0,1,1,1,1,3,3,3,3,4,4,4,4,1,1,1,1,3,3,3,3
288: 名前SAdjust:
289:         .dc.b   'SAdjust.r',0
290: SXBASIC_GCOLOR送信メッセージ:
291:         .dc.b   'GMODE 24',0
292: グラフィックモードチェックエラーメッセージ:
293:         .dc.b   'GV-RAM が 65536 色モードではありません。',0
294:         .even
295:
296: ******************************************
297: 初期化ルーチン:
298: 保存レジスタ    reg     d1-d7/a1-a5
299:         movem.l 保存レジスタ,-(sp)
300:         lea     初期化ヘッダ(a5),a0
301:         moveq   #$00,d0
302:         move.w  #(初期化テイル-初期化ヘッダ)/2-1,d1
303: ワーククリアループ:
304:         move.w  d0,(a0)+
305:         dbra    d1,ワーククリアループ
306:         move.w  #128,スイッチA値(a5)
307:         .dc.w   $FE0E
308:         lsr.w   #2,d0
309:         and.w   #510,d0
310:         move.w  d0,スイッチS値+ptX(a5)
311:         .dc.w   $FE0E
312:         lsr.w   #2,d0
313:         and.w   #510,d0
314:         move.w  d0,スイッチS値+ptY(a5)
315:         move.w  #$0001,-(sp)
316:         clr.l   -(sp)
317:         clr.l   -(sp)
318:         move.l  コマンドラインのアドレス(a5),-(sp)
319:         SX      __TSTakeParam
320:         lea     14(sp),sp
321:         bmi     初期化終了
322:         movea.l a0,a2
323:         pea     (a2)
324:         movea.l sp,a0
325:         bsr     コマンドライン解析ルーチン
326:         SX      __MMDisposePtr
327:         addq.l  #4,sp
328:         moveq   #$00,d0
329: 初期化終了:
330:         movem.l (sp)+,保存レジスタ
331:         rts
332:
333: ******************************************
334: コマンドライン解析ルーチン:
335: 保存レジスタ    reg     d1-d7/a1-a5
336:         movem.l 保存レジスタ,-(sp)
337:         movea.l a0,a4
338:         movea.l $0000(a4),a2
339:         move.l  (a2)+,d2
340: コマンドライン解析01:
341:         subq.l  #1,d2
342:         bmi     コマンドライン解析終了
343:         movea.l (a2)+,a1
344:         move.b  (a1),d0
345:         beq     コマンドライン解析01
346:         cmpi.b  #'-',d0
347:         beq     コマンドライン解析02
348:         cmpi.b  #'/',d0
349:         bne     コマンドライン解析ff
350: コマンドライン解析02:
351:         addq.l  #1,a1
352:         move.b  (a1)+,d0
353:         and.b   #$5f,d0
354:         cmpi.b  #'G',d0
355:         bne     コマンドライン解析03
356:         move.w  #$0001,スイッチGフラグ(a5)
357:         bra     コマンドライン解析ff
358: コマンドライン解析03:
359:         cmpi.b  #'M',d0
360:         bne     コマンドライン解析04
361:         move.w  #$0001,スイッチMフラグ(a5)
362:         bra     コマンドライン解析ff
363: コマンドライン解析04:
364:         cmpi.b  #'S',d0
365:         bne     コマンドライン解析05
366:         pea     (a1)
367:         SX      $a3df
368:         addq.l  #4,sp
369:         move.w  d0,スイッチS値+ptX(a5)
370:         pea     1(a0)
371:         SX      $a3df
372:         addq.l  #4,sp
373:         move.w  d0,スイッチS値+ptY(a5)
374:         bra     コマンドライン解析ff
375: コマンドライン解析05:
376:         cmpi.b  #'A',d0
377:         bne     コマンドライン解析06
378:         pea     (a1)
379:         SX      $a3df
380:         addq.l  #4,sp
381:         move.w  d0,スイッチA値(a5)
382:         bra     コマンドライン解析ff
383: コマンドライン解析06:
384:         cmpi.b  #'K',d0
385:         bne     コマンドライン解析07
386:         move.w  #1,スイッチKフラグ(a5)
387:         pea     (a1)
388:         SX      $a3df
389:         addq.l  #4,sp
390:         move.w  d0,スイッチK値(a5)
391:         bra     コマンドライン解析ff
392: コマンドライン解析07:
393: コマンドライン解析ff:
394:         bra     コマンドライン解析01
395: コマンドライン解析終了:
396:         movem.l (sp)+,保存レジスタ
397:         rts
398:
399: ******************************************
400: 正常終了処理:
401:         moveq   #$00,d0
402: 起動失敗による終了処理:
403:         move.l  d0,d2
404: 終了処理終了:
405:         move.l  d2,-(sp)
406:         SX      __TSExit
407:
408: ******************************************
409: 描画ルーチン:
410: 保存レジスタ    reg     d0-d5/a0-a1
411:         movem.l 保存レジスタ,-(sp)
412:         clr.l -(sp)
413:         DOS     _SUPER
414:         move.l  d0,(sp)
415:         bsr     全クリアルーチン
416:         moveq   #0,d0
417:         moveq   #0,d1
418:         lea     座標テーブル(pc),a1
419:         move.w  #256-1,d4
420: 描画ループ:
421:         move.w  (a1)+,d0
422:         move.w  (a1)+,d1
423:         move.l  d0,d5
424:         sub.l   #50,d5
425:         lsl.l   #6,d5
426:         add.w   d2,d0
427:         add.w   d3,d1
428:         pea     玉パターン(pc)
429:         move.l  #$00010000,-(sp)
430:         sub.l   d5,(sp)
431:         move.w  d1,-(sp)
432:         move.w  d0,-(sp)
433:         movea.l sp,a0
434:         bsr     玉描画ルーチン
435:         lea     12(sp),sp
436:         add.w   スイッチS値+ptX(a5),d2
437:         add.w   スイッチS値+ptY(a5),d3
438:         dbra    d4,描画ループ
439:         DOS     _SUPER
440:         addq.l  #4,sp
441:         moveq   #0,d0
442:         movem.l (sp)+,保存レジスタ
443:         rts
444: 座標テーブル:
445: .dc.w455,252
446: .dc.w455,247
447: .dc.w455,242
448: .dc.w455,237
449: .dc.w454,232
450: .dc.w453,227
451: .dc.w453,222
452: .dc.w452,217
453: .dc.w451,213
454: .dc.w450,208
455: .dc.w448,203
456: .dc.w447,198
457: .dc.w445,194
458: .dc.w444,189
459: .dc.w442,185
460: .dc.w440,180
461: .dc.w438,175
462: .dc.w436,171
463: .dc.w434,167
464: .dc.w432,162
465: .dc.w430,158
466: .dc.w427,154
467: .dc.w424,150
468: .dc.w422,145
469: .dc.w419,141
470: .dc.w416,137
471: .dc.w413,133
472: .dc.w410,130
473: .dc.w407,126
474: .dc.w404,122
475: .dc.w400,119
476: .dc.w397,115
477: .dc.w393,112
478: .dc.w390,108
479: .dc.w386,105
480: .dc.w382,102
481: .dc.w379,99
482: .dc.w375,96
483: .dc.w371,93
484: .dc.w367,90
485: .dc.w362,88
486: .dc.w358,85
487: .dc.w354,82
488: .dc.w350,80
489: .dc.w345,78
490: .dc.w341,76
```

▶DōGA CGAシステムの最新バージョンも配布してほしい。　井戸　浩登(32)東京都

```
491: .dc.w337,74
492: .dc.w332,72
493: .dc.w327,70
494: .dc.w323,68
495: .dc.w318,67
496: .dc.w314,65
497: .dc.w309,64
498: .dc.w304,62
499: .dc.w299,61
500: .dc.w295,60
501: .dc.w290,59
502: .dc.w285,59
503: .dc.w280,58
504: .dc.w275,57
505: .dc.w270,57
506: .dc.w265,57
507: .dc.w260,57
508: .dc.w256,57
509: .dc.w252,57
510: .dc.w247,57
511: .dc.w242,57
512: .dc.w237,57
513: .dc.w232,58
514: .dc.w227,59
515: .dc.w222,59
516: .dc.w217,60
517: .dc.w213,61
518: .dc.w208,62
519: .dc.w203,64
520: .dc.w198,65
521: .dc.w194,67
522: .dc.w189,68
523: .dc.w185,70
524: .dc.w180,72
525: .dc.w175,74
526: .dc.w171,76
527: .dc.w167,78
528: .dc.w162,80
529: .dc.w158,82
530: .dc.w154,85
531: .dc.w150,88
532: .dc.w145,90
533: .dc.w141,93
534: .dc.w137,96
535: .dc.w133,99
536: .dc.w130,102
537: .dc.w126,105
538: .dc.w122,108
539: .dc.w119,112
540: .dc.w115,115
541: .dc.w112,119
542: .dc.w108,122
543: .dc.w105,126
544: .dc.w102,130
545: .dc.w99,133
546: .dc.w96,137
547: .dc.w93,141
548: .dc.w90,145
549: .dc.w88,150
550: .dc.w85,154
551: .dc.w82,158
552: .dc.w80,162
553: .dc.w78,167
554: .dc.w76,171
555: .dc.w74,175
556: .dc.w72,180
557: .dc.w70,185
558: .dc.w68,189
559: .dc.w67,194
560: .dc.w65,198
561: .dc.w64,203
562: .dc.w62,208
563: .dc.w61,213
564: .dc.w60,217
565: .dc.w59,222
566: .dc.w59,227
567: .dc.w58,232
568: .dc.w57,237
569: .dc.w57,242
570: .dc.w57,247
571: .dc.w57,252
572: .dc.w57,256
573: .dc.w57,260
574: .dc.w57,265
575: .dc.w57,270
576: .dc.w57,275
577: .dc.w58,280
578: .dc.w59,285
579: .dc.w59,290
580: .dc.w60,295
581: .dc.w61,299
582: .dc.w62,304
583: .dc.w64,309
584: .dc.w65,314
585: .dc.w67,318
586: .dc.w68,323
587: .dc.w70,327
588: .dc.w72,332
589: .dc.w74,337
590: .dc.w76,341
591: .dc.w78,345
592: .dc.w80,350
593: .dc.w82,354
594: .dc.w85,358
595: .dc.w88,362
596: .dc.w90,367
597: .dc.w93,371
598: .dc.w96,375
599: .dc.w99,379
600: .dc.w102,382
601: .dc.w105,386
602: .dc.w108,390
603: .dc.w112,393
604: .dc.w115,397
605: .dc.w119,400
606: .dc.w122,404
607: .dc.w126,407
608: .dc.w130,410
609: .dc.w133,413
610: .dc.w137,416
611: .dc.w141,419
612: .dc.w145,422
613: .dc.w150,424
614: .dc.w154,427
615: .dc.w158,430
616: .dc.w162,432
617: .dc.w167,434
618: .dc.w171,436
619: .dc.w175,438
620: .dc.w180,440
621: .dc.w185,442
622: .dc.w189,444
623: .dc.w194,445
624: .dc.w198,447
625: .dc.w203,448
626: .dc.w208,450
627: .dc.w213,451
628: .dc.w217,452
629: .dc.w222,453
630: .dc.w227,453
631: .dc.w232,454
632: .dc.w237,455
633: .dc.w242,455
634: .dc.w247,455
635: .dc.w252,455
636: .dc.w256,455
637: .dc.w260,455
638: .dc.w265,455
639: .dc.w270,455
640: .dc.w275,455
641: .dc.w280,454
642: .dc.w285,453
643: .dc.w290,453
644: .dc.w295,452
645: .dc.w299,451
646: .dc.w304,450
647: .dc.w309,448
648: .dc.w314,447
649: .dc.w318,445
650: .dc.w323,444
651: .dc.w327,442
652: .dc.w332,440
653: .dc.w337,438
654: .dc.w341,436
655: .dc.w345,434
656: .dc.w350,432
657: .dc.w354,430
658: .dc.w358,427
659: .dc.w362,424
660: .dc.w367,422
661: .dc.w371,419
662: .dc.w375,416
663: .dc.w379,413
664: .dc.w382,410
665: .dc.w386,407
666: .dc.w390,404
667: .dc.w393,400
668: .dc.w397,397
669: .dc.w400,393
670: .dc.w404,390
671: .dc.w407,386
672: .dc.w410,382
673: .dc.w413,379
674: .dc.w416,375
675: .dc.w419,371
676: .dc.w422,367
677: .dc.w424,362
678: .dc.w427,358
679: .dc.w430,354
680: .dc.w432,350
681: .dc.w434,345
682: .dc.w436,341
683: .dc.w438,337
684: .dc.w440,332
685: .dc.w442,327
686: .dc.w444,323
687: .dc.w445,318
688: .dc.w447,314
689: .dc.w448,309
690: .dc.w450,304
691: .dc.w451,299
692: .dc.w452,295
693: .dc.w453,290
694: .dc.w453,285
695: .dc.w454,280
696: .dc.w455,275
697: .dc.w455,270
698: .dc.w455,265
699: .dc.w455,260
700: .dc.w456,256
701: 玉パターン:
702: .dc.l$00000000,$00000000,$0016002a,$003a0038,$0032002a,$00180008,$000000
00,$00000000
703: .dc.l$00000000,$00080032,$003e003e,$087e003a,$00360030,$002c0024,$001200
02,$00000000
704: .dc.l$0000000c,$003a087e,$10be18fe,$10be087c,$00360032,$002c0026,$001e00
14,$00000000
705: .dc.l$00000032,$10be31be,$52be52be,$39fe18fe,$08780032,$002c0026,$002000
1a,$000e0000
706: .dc.l$0016087e,$31be7bfe,$a53e94be,$633e31be,$08780032,$002a0026,$001e00
1a,$00180004
707: .dc.l$003210be,$633eb5be,$e73ebdfe,$7bfe39fe,$18fa0030,$002a0024,$001e00
18,$001a000e
708: .dc.l$003e18fe,$5afeb5be,$defebdfe,$73be39fe,$10b6086e,$00280022,$001c00
1a,$00180014
709: .dc.l$003e10be,$423e843e,$a53e8c7e,$5afe297c,$10b4002c,$00260020,$001c00
18,$001a0018
710: .dc.l$003c087e,$213e4a7e,$5afe52be,$31bc18f8,$002e002a,$0022001e,$001800
1a,$0018001a
711: .dc.l$003e003e,$10be213e,$297e213c,$18f80870,$086c0026,$0022001a,$001a00
18,$001a0014
712: .dc.l$002e003c,$003a087c,$08780876,$0030002c,$00260022,$001c001a,$001800
1a,$0018000e
713: .dc.l$00140036,$00380034,$08740030,$002c0028,$0024001e,$001a0018,$001a00
18,$001a0002
714: .dc.l$00000026,$00300030,$002c002c,$00260024,$001e001a,$0018001a,$001800
1a,$000e0000
715: .dc.l$00000008,$0028002a,$00280026,$0022001e,$001a0018,$001a0018,$001a00
14,$00020000
716: .dc.l$00000000,$0004001c,$00200020,$001c001a,$0018001a,$0018001a,$001000
02,$00000000
717: .dc.l$00000000,$00000000,$000a0012,$001a0018,$001a0018,$00100006,$000000
00,$00000000
718:
719: ****************************************
720: 玉描画ルーチン:
721: *        引き数 (a0)
722: *                  .dc.w   X座標
723: *                  .dc.w   Y座標
724: *                  .dc.l   縮小率
725: *                  .dc.l   パターンアドレス
726: 保存レジスタ     reg      d0-d7/a0-a1/a4
727:         movem.l 保存レジスタ,-(sp)
728:         movea.l a0,a4
729:         movea.l 8(a4),a1
730:         move.l  4(a4),d7
731:         move.l  (a4),d0
732:         moveq   #0,d1
733:         move.w  d0,d1
734:         clr.w   d0
735:         swap    d0
736:         move.l  d0,d6
737:         move.w  #16-1,d3
738: 小玉描画ループY:
```

▶SX-WINDOWまわりがよくなってきました。あと，「書院 SX-68K」も出るといいな。
新谷　貴幸(19)青森県

```
739:            swap    d1
740:            add.l   d7,d1
741:            swap    d1
742:            move.w  #16-1,d2
743: 小玉描画ループX:
744:            swap    d0
745:            add.l   d7,d0
746:            swap    d0
747:            move.w  (a1)+,-(sp)
748:            move.w  d1,-(sp)
749:            move.w  d0,-(sp)
750:            movea.l sp,a0
751:            bsr     点描画ルーチン
752:            addq.l  #6,sp
753:            dbra    d2,小玉描画ループX
754:            move.l  d6,d0
755:            dbra    d3,小玉描画ループY
756:            movem.l (sp)+,保存レジスタ
757:            rts
758:
759: ******************************************
760: 点描画ルーチン:
761: *          引き数 (a0)
762: *                  .dc.w    X座標
763: *                  .dc.w    Y座標
764: *                  .dc.w    色
765: 保存レジスタ    reg     d0/a0-a1
766:            tst.w   4(a0)
767:            beq     点描画リターン
768:            movem.l 保存レジスタ,-(sp)
769:            lea     $c00000,a1
770:            moveq   #0,d0
771:            move.w  (a0)+,d0
772:            and.w   #511,d0
773:            add.w   d0,d0
774:            add.w   d0,a1
775:            move.w  (a0)+,d0
776:            and.w   #511,d0
777:            lsl.l   #8,d0
778:            add.l   d0,d0
779:            add.l   d0,d0
780:            add.l   d0,a1
781:            move.w  (a0),(a1)
782:            movem.l (sp)+,保存レジスタ
783: 点描画リターン:
784:            rts
785:
786: *****************************************
787: 全クリアルーチン:
788: 保存レジスタ    reg     d0-d1/a0
789:            movem.l 保存レジスタ,-(sp)
790:            lea     $c00000,a0
791:            moveq   #0,d0
792:            moveq   #-1,d1
793: 全クリア01:
794:            move.l  d0,(a0)+
795:            move.l  d0,(a0)+
796:            dbra    d1,全クリア01
797:            movem.l (sp)+,保存レジスタ
798:            rts
799:
800: ***********************************************
801: モジュールテイル:
802:
803:            .end    コマンドラインスタート
```

リスト3　壁紙動画.LZH

(1477バイトでセーブ)

```
000000  23 09 2D 6C 68 35 2D 9F : 2E
000008  05 00 00 92 08 00 00 62 : 01
000010  98 C1 1C 20 01 0A 95 C7 : FC
000018  8E 86 93 AE 89 E6 2E 72 : 64
000020  02 5C 48 00 00 05 0F 73 : 2D
000028  A2 EE E8 D3 5D 27 FB D4 : 9E
000030  A1 2E 1B 86 9A 38 81 51 : 14
000038  37 35 B2 C2 50 BD 6E DE : 39
000040  06 33 06 D6 DE 03 3C 5C : 8E
000048  1D 69 6D 2E 26 C4 0A 31 : 46
000050  7B B3 8C 10 8C 0D 18 42 : BD
000058  28 51 84 D9 91 69 0D B3 : 90
000060  B4 7C 5B FF FB 62 49 91 : C1
000068  D0 58 AC 99 9B C1 66 76 : A5
000070  55 F1 E8 A6 7C 0C 64 DB : 9B
000078  C0 CD 94 81 51 66 9A 12 : 05
-----------------------------------
CKSUM:  29 2F DF 93 C5 18 01 26 86BC

000080  C0 BD F3 F7 06 36 33 97 : 6D
000088  C0 C7 C0 F9 D9 1D F7 68 : 95
000090  DB 6D BB E8 28 C0 86 44 : 9D
000098  CB 28 88 9B 11 52 F3 4B : B7
0000A0  C5 FC B1 39 B3 AC 80 4C : D6
0000A8  6B 37 E4 CB 77 8D 01 F5 : 4B
0000B0  E9 50 37 BC 26 17 03 BA : 26
0000B8  94 AF 89 9B D7 45 55 2B : 03
0000C0  81 73 26 F6 9E C2 71 54 : 35
0000C8  BC 74 4C E6 62 17 8F D6 : 40
0000D0  7C F3 72 44 9F C2 45 1F : EA
0000D8  EF 83 A3 4B CD 8F 9C 47 : 9F
0000E0  FF 7F 9B 0F C3 B4 F0 82 : 11
0000E8  DB 5C DC 42 39 5A 41 CE : F7
0000F0  E1 5B AC DC 43 98 2B 75 : 3F
0000F8  1F 83 78 54 3F 8A A4 C3 : 9E
-----------------------------------
CKSUM:  55 61 6D BA 29 54 5D CC 19B5

000100  31 9F C0 20 EC 52 A6 51 : E5
000108  9A 58 65 4C F6 BC 8E 97 : 7A
000110  61 73 26 3E EE B3 13 58 : 44
000118  BC 7F BB 1F D5 C7 4F 1E : 1E
000120  EB 1F 07 1F D0 C7 4E 47 : 5C
000128  3E 4F A3 CB FF D2 3E 49 : 53
000130  1D 49 1F FE 47 52 4D 9F : 08
000138  33 A3 23 AB 23 DF 91 F8 : 2F
000140  48 E8 C8 FD 71 91 22 BC : D5
000148  26 39 49 E1 36 BE 2A 46 : ED
000150  D6 CF E7 12 97 85 45 EC : EB
000158  AB F1 80 6F 00 38 AA F3 : 60
000160  C0 3C 20 0F 19 5F 58 03 : FE
000168  18 03 C5 54 8B 8E 03 82 : D2
000170  03 8E 03 CC 01 C0 01 C5 : E7
000178  01 F4 C0 76 00 34 80 73 : 52
-----------------------------------
CKSUM:  2C E5 12 60 C1 3F 17 23 5833

000180  00 75 E0 3A F0 1F 48 07 : ED
000188  E7 00 DE 00 FB 31 F2 6D : 50
000190  AD EC E2 95 92 76 25 C0 : FD
000198  7E 00 4A 0D AD 91 44 B0 : 07
0001A0  87 06 D2 D0 A2 45 83 08 : A1
0001A8  AD 60 C6 B2 28 D6 50 E2 : B5
0001B0  59 C1 4C 11 7F 0E 2D BC : ED
0001B8  22 81 4A AB 34 AD A5 76 : 94
0001C0  2E 24 EF 8B C9 7C 4E A3 : 02
0001C8  92 D0 01 36 BA C4 EB 9F : A1
0001D0  13 68 0D 02 0F 0C 8E D7 : 0A
0001D8  1D BA 39 6B 1E 86 28 7A : C1
0001E0  D3 AC 7D C4 90 9E D7 C2 : 87
0001E8  9E E3 C4 9F 57 1C A7 E9 : E7
0001F0  D7 0A EE 48 00 66 A3 7C : 9C
0001F8  BF 7D A1 47 FD D4 72 CD : 34
-----------------------------------
CKSUM:  B8 35 1E 3A 3B F3 CA 87 35E2

000200  59 5E 90 AD ED 0F 13 BB : BE
000208  19 70 7F 42 66 0E FD 5C : 17
000210  BF A7 DB C8 E4 F4 39 1D : 37
000218  A3 C4 5C 86 E5 07 8F 0D : D1
000220  13 EF 53 8D 3C 9E D7 AC : 3F
000228  B2 57 F3 CA 57 0A E6 4E : 5B
000230  B3 CE 06 2F 11 93 44 16 : B4
000238  EB D1 91 1D 30 33 D8 20 : C5
000240  53 D6 EB E3 02 3B 09 77 : B4
000248  37 32 53 BA 0D 7C 04 BB : BE
000250  7E BE 23 FD 12 67 80 5B : B0
000258  AA F2 EA BC 99 6D F5 AD : EA
000260  A9 2B F3 18 13 E3 7A 44 : 93
000268  59 07 B2 5C 1C 4A EC 4D : 0D
000270  32 2A A1 7D DA 0B EE E0 : 2D
000278  47 63 29 0D 14 A3 96 8E : BB
-----------------------------------
CKSUM:  64 95 DD 34 C7 EC 1D AA 28A4

000280  5B B7 44 C6 56 78 89 E9 : 5C
000288  2F 53 6E 92 50 CD 1B 97 : 51
000290  3F 9D 46 20 7B E2 3D 4B : 27
000298  18 16 EA 5C BF 9D 43 CD : E0
0002A0  3F DA 34 E6 76 55 84 EA : 6C
0002A8  1C 8B 29 D4 3E C3 6A 3F : 4E
0002B0  3F 44 EE 57 18 E7 6D 7E : B2
0002B8  DE 63 DA 89 7E 65 CC 97 : EA
0002C0  13 22 63 A9 A4 96 1D 0B : A3
0002C8  1E 77 0E E2 4E 01 5A 76 : A4
0002D0  37 B3 D3 C2 C0 5A 3F 77 : 4F
0002D8  04 9E 6D D5 84 F4 91 54 : 41
0002E0  CF 57 75 E6 C7 F4 72 E5 : 93
0002E8  ED CC F5 C1 28 FE 6E 01 : 04
0002F0  37 28 17 C7 65 2B 5A 96 : BD
0002F8  1B 1F A3 9D E9 DC 49 0F : 97
-----------------------------------
CKSUM:  D3 1D DC 9B 9D 06 15 AD 24EF

000300  17 08 E4 87 5B 5B 20 E5 : 45
000308  C1 0E 5D 67 1C 49 0A 77 : 79
000310  25 BB 76 52 D0 B8 92 ED : AF
000318  84 A8 66 9F EA D2 6F 15 : 71
000320  BB 58 92 2F 48 EC F6 DC : DA
000328  16 94 AA F7 19 91 3A 55 : 84
000330  90 E5 DA 84 77 D9 0F 90 : C2
000338  D2 A0 54 7B B0 3C 35 28 : 8A
000340  D7 B7 27 C7 23 BE 60 3D : FA
000348  AA 88 3B DA C0 B7 55 91 : A4
000350  FB B6 3F 2A AC 1C BE EF : 8F
000358  D8 11 AB A8 51 BE 16 04 : 65
000360  27 8B EA 83 9B B0 A7 90 : A1
000368  14 D2 51 CE C2 9C 8E FA : EB
000370  99 B1 67 67 15 57 4F 8C : 5F
000378  13 E2 78 13 E1 F8 53 DC : 88
-----------------------------------
CKSUM:  EF E0 ED 42 EC AA FF FA 2E9C

000380  6C E2 AA E9 EE 02 7B F5 : 41
000388  0F D8 4F 6D DF 4F D6 CC : 73
000390  D5 D1 48 22 EF D4 4D 38 : 58
000398  DA B3 DE 6D D4 62 F0 60 : 5E
0003A0  AE F1 F6 AC F7 A8 6A CF : 19
0003A8  FF B1 27 BC 97 AF 89 25 : 87
0003B0  98 1A 76 2A 11 4B EF 3E : DB
0003B8  E8 3D E3 3E D7 A5 AF 07 : 78
0003C0  8B 7D D2 56 D2 F6 D0 A4 : 6C
0003C8  8C 6B E4 B6 64 8B 41 5B : 1C
0003D0  69 1A 6A 11 E9 42 76 D4 : 73
0003D8  C5 E3 CD 3D 96 49 C2 21 : 74
0003E0  CA 32 5A 39 63 95 C1 17 : 5F
0003E8  BC A5 44 17 9F 6D 03 16 : E1
0003F0  F5 11 18 E8 08 A7 40 3F : 34
0003F8  74 12 BF D5 9A 68 A5 94 : 55
-----------------------------------
CKSUM:  8B 16 F7 1C 5F EB 11 86 8660

000400  D1 E4 CF E7 D8 22 20 F4 : 79
000408  7A 0B 69 0F BA 7B C1 E2 : D5
000410  9D 6D 44 AF D6 9A 63 B2 : 82
000418  E9 CD 14 89 FD 88 CF 7F : 26
000420  23 5B 16 FF 8D 3F 7F 36 : 14
000428  7F 09 BB 21 0B BA 1F 6D : B5
000430  4D 70 52 98 EC 92 85 2B : D5
000438  B3 0A 49 A2 DD 73 29 03 : 24
000440  2D A6 74 88 A8 16 F0 1D : 9A
000448  AD 05 5B FF 25 7B CA 71 : E7
000450  01 D7 1E 16 69 93 EA FF : F1
000458  A6 FB F0 66 B7 7C 5F D1 : 5A
000460  BD F8 A3 6B 93 1A BA F1 : 1B
000468  53 FC 06 6C 52 84 6A 62 : 63
000470  6A 9A B8 3C 93 02 2A AB : 62
000478  0F 29 F4 FE 0A 2D 3F 7D : 1D
-----------------------------------
CKSUM:  7D 3B 2E 9C 35 2A EF B1 9222

000480  FB 4F 08 D9 F3 61 38 E9 : A0
000488  D6 53 8E D3 6D 47 82 70 : 30
000490  D1 F6 D1 64 E4 E3 66 FB : 24
000498  AA 33 1A 4B A5 B3 AF E6 : 2F
0004A0  A4 EE D9 6C D2 97 88 2D : F5
0004A8  D3 3E EC CC 74 E3 2D 9C : E9
0004B0  97 6D F7 1E AD 23 E2 B6 : 81
0004B8  65 02 9D 4D 40 F3 C6 74 : BE
0004C0  F9 6C E3 E2 BA 49 6C F7 : 90
0004C8  F3 56 CF 1D E9 6F 07 A5 : 39
0004D0  D5 B5 20 71 4A 89 54 47 : 89
0004D8  83 54 16 CE 7B 5A 4C 5E : 3A
0004E0  D7 D0 13 D4 16 D4 03 E9 : 64
0004E8  8F BA 52 F7 A2 F5 D3 7F : 7B
0004F0  87 EB 8C 7E 3D 37 A9 B4 : 4D
0004F8  5E EB 84 13 5D 71 94 EC : 2E
-----------------------------------
CKSUM:  4E 91 37 98 D6 DA 52 76 5B20

000500  47 8B 99 AD 8B E2 15 FC : 96
000508  98 28 DC FA D2 64 AD 51 : CA
000510  EF 9E 74 16 9B E5 DF AD : 23
000518  A4 4C 07 3D C2 A2 1C E9 : 9D
000520  95 18 E7 50 A9 0F 3A D3 : A9
000528  B1 4A C0 D2 1B BD A3 4A : 52
000530  75 68 B3 6A 14 AA 8F DD : 24
000538  69 50 AD 50 CC BD AA B4 : 9D
000540  E3 9D A2 BB 61 C9 EA B3 : A4
000548  1C B5 56 8B 01 66 D7 69 : 59
000550  E1 1F 76 D2 A1 74 1D 35 : AF
000558  D0 7F 65 D0 74 22 72 30 : BC
000560  08 32 59 1F 86 59 87 D5 : ED
000568  9D CA 3E EC 79 C5 A2 76 : E7
000570  62 64 EF 6A E7 0C BD C9 : 98
000578  6E D2 7D D6 BE 1B CA 6E : A4
-----------------------------------
CKSUM:  BB D9 CD 09 79 0A D3 94 B3FD

000580  5B F9 93 0B F2 2A 75 F9 : 7C
000588  A9 5F 93 D7 E4 6F B9 FB : 79
000590  84 F0 82 7A 90 34 47 8D : 08
000598  18 C2 F3 A0 A3 91 82 2E : 51
0005A0  72 6F 8A B4 6F 4B 9E 25 : 9C
0005A8  4F 58 79 81 30 27 BA D7 : 89
0005B0  66 B8 5F 9A BA C1 E1 0F : 82
0005B8  FD 72 A7 85 2D 05 96 5A : BD
0005C0  29 BF C3 88 00 00 00 00 : 33
0005C8  00 00 00 00 00 00 00 00 : 00
0005D0  00 00 00 00 00 00 00 00 : 00
0005D8  00 00 00 00 00 00 00 00 : 00
0005E0  00 00 00 00 00 00 00 00 : 00
0005E8  00 00 00 00 00 00 00 00 : 00
0005F0  00 00 00 00 00 00 00 00 : 00
0005F8  00 00 00 00 00 00 00 00 : 00
-----------------------------------
CKSUM:  ED BA 67 D8 8F 96 C6 14 E7FD
```

▶毎月の特集には興味をひかれます。特集を読むと毎月やりたいことが増えて困ります。CGやコンピュータミュージック，C言語もやりたいし。でも，その前にメモリとハードディスクが欲しい。
工藤　大輔(19)北海道

リスト4 壁動玉々.LZH

(1952バイトでセーブ)

```
000000  23 17 2D 6C 68 35 2D 7A : 17
000008  07 00 00 E6 0A 00 00 79 : 70
000010  19 D5 06 20 01 0A 95 C7 : 7B
000018  93 AE 8B CA 81 58 2E 72 : 0F
000020  58 A0 48 00 00 06 FE 73 : B7
000028  EE BB 85 69 47 36 FD FD : 0E
000030  2C 27 96 12 EF 61 23 2A : 98
000038  CE 59 52 31 4E 14 0A 99 : AF
000040  59 C8 D2 96 D0 C1 ED 51 : 58
000048  A8 55 18 27 2B C0 BA 51 : 32
000050  44 46 C0 E7 8E 82 4B 86 : 12
000058  F6 5E C4 EC 70 39 79 74 : 9A
000060  36 58 C4 79 19 89 04 8C : FD
000068  16 2B 12 B6 0A AF 72 F6 : 2A
000070  D0 8E CE 91 91 8B B6 7A : 09
000078  72 46 03 BB 31 91 73 2B : D6
-----------------------------------
CKSUM:  DF 8D 88 F3 56 D8 22 22 5B13

000080  7D FB D4 A2 F7 6D A6 F8 : F0
000088  9B E8 2B 6E E9 6B 8D 37 : 34
000090  7B 56 ED E3 E0 32 DF 60 : F2
000098  02 5C 05 73 43 FE 7C B8 : 4B
0000A0  99 FC 9A 17 81 96 87 52 : 36
0000A8  14 9E DE 3C 73 B7 12 5C : 64
0000B0  FA C1 94 C3 3C F0 2B 02 : 6B
0000B8  77 8F 3E 51 E7 75 B7 52 : FA
0000C0  B0 49 72 2A C4 97 EB AC : 87
0000C8  64 78 FE A7 BA E8 B8 CB : A6
0000D0  78 BF 35 AB AF 24 5C 7C : C2
0000D8  F2 8B 53 2D 68 93 BC 89 : 3D
0000E0  5A 6B 95 62 86 A7 33 47 : 63
0000E8  3A C6 D3 42 56 27 48 04 : DE
0000F0  5E 70 2D 4D 79 FD 23 AD : 8E
0000F8  D1 14 B2 DF F5 2E 4E CD : B4
-----------------------------------
CKSUM:  F4 3F 7A 46 F9 E9 B0 8A 1120

000100  D1 5A AF FE 5B A9 9C 52 : CA
000108  81 BA EB AD 61 18 5E 7B : 25
000110  BB 0B 96 BE F4 00 AC DA : 94
000118  FD C7 9B 81 98 AD 64 AC : 35
000120  D0 CC C4 6E 13 27 6E C0 : 36
000128  4C 72 BF F8 4D AA 97 59 : 5C
000130  4C DB 98 B8 CE D5 2C 5E : A4
000138  CB BB D5 AC 09 BE D0 44 : E2
000140  A0 6E 05 E6 B3 57 FD C4 : C4
000148  AB CB 6D BC C4 FB 9F 56 : 53
000150  33 BB FF 57 DA 73 E8 9F : 18
000158  B5 03 D5 15 6B 0F 57 92 : 05
000160  82 6F D5 8D C8 69 B0 B9 : ED
000168  80 29 7C 0E 23 3D D9 3D : A9
000170  DD 11 0B 37 F2 87 7F 9F : C7
000178  3B F4 E4 50 95 6B D5 FF : 37
-----------------------------------
CKSUM:  8A 4E 41 E4 AD 3E C3 ED F19F

000180  69 AD 53 66 B3 4B 36 D1 : D4
000188  36 0D 66 A8 BE 6E C7 02 : 46
000190  FF E4 EC EF 2A 66 EC 47 : 81
000198  CD D8 F9 77 DC 0B FB 7F : 76
0001A0  B2 6E C5 AC 2F AB E4 7B : CA
0001A8  C0 BF 85 F4 FC BC FE 0D : BB
0001B0  94 2E 6C 2E F1 CE 16 F1 : 22
0001B8  D8 7B AE 8C 18 8F BF 7C : 6F
0001C0  0D 80 BD 00 2E 1C E5 E7 : 60
0001C8  9B 61 72 66 3A B8 16 0B : E7
0001D0  EC 1F E3 7B 00 65 E3 E6 : 97
0001D8  F6 36 44 7D FF 23 46 D9 : 2E
0001E0  47 5E 91 6A AB 60 30 1B : F6
0001E8  89 2A D7 C4 53 29 5B AD : D2
0001F0  8E D7 B4 16 D2 19 74 4F : DD
0001F8  75 CB E2 CF B9 5F 14 EA : 07
-----------------------------------
CKSUM:  A6 AC 56 3F 9B 4B D2 40 A53E

000200  FE 09 A5 97 F1 BA F2 2F : 0F
000208  E3 3C FE D9 FB 54 CA 2F : 3E
000210  16 C7 2A B5 82 E4 EA BE : CA
000218  2B 07 57 96 D5 CC A9 FE : 67
000220  95 FC 8F 76 4E CB 2E AF : 8C
000228  F6 C8 00 FB 20 1C FF DD : D1
000230  00 FE B6 47 E6 2A 1D 0F : 37
000238  D2 47 E6 3B F2 7A F5 8E : 29
000240  BF 2D 76 EA CB DF 22 CA : E2
000248  4D 56 AE 8B F1 B5 9D D2 : F1
000250  0A 4D 74 D3 E7 5A A6 B0 : 35
000258  BF 73 5D 36 97 4B F8 B4 : 53
000260  CD 6E A8 FE 28 5B CF 77 : AA
000268  E2 DA C2 EB 42 FD F0 B8 : 50
000270  D0 B1 4C 21 E8 42 FF 06 : 1D
000278  10 C7 A9 BC 71 29 00 4E : 24
-----------------------------------
CKSUM:  E3 1F A3 F2 86 45 A9 C6 6B9E

000280  B7 12 6D A2 10 8C 5D 45 : 16
000288  A8 B6 0A C6 67 F5 AD 12 : 49
000290  D5 23 6A 1C 89 5F BB AE : CF
000298  59 80 AB BB D7 77 B3 93 : D3
0002A0  41 95 E7 8C CE 16 7B 76 : 1E
0002A8  79 E5 96 7C D7 57 59 29 : 20
0002B0  C6 CB 54 E4 CC DD D3 0D : 52
0002B8  99 85 B3 DA DC F6 6B 59 : 41
0002C0  0B 6E 04 29 7A A5 7F 0D : 51
0002C8  6F 9A 50 09 AA D9 33 C6 : DE
0002D0  71 FD 73 EB 75 58 A8 9F : E0
0002D8  78 F6 D2 3F 50 BD 50 C0 : 9C
0002E0  7D 69 ED D5 48 ED 67 57 : 9B
0002E8  46 7B 75 8F DA 2D 4F 79 : 94
0002F0  E0 A9 B9 60 12 B6 D4 24 : 62
0002F8  8E B9 6A B9 82 B5 D1 85 : F7
-----------------------------------
CKSUM:  3A 76 2E DE C3 AF 8F 48 1143

000300  79 2B 55 AB CC 14 3A 97 : 55
000308  EA C7 DE E6 3D 69 DA 55 : 4A
000310  85 ED 72 02 B0 06 78 4E : 62
000318  65 2F 1F 2D E2 8B 11 DD : 3B
000320  EA ED 66 ED AB 04 9F F0 : 68
000328  80 2C 45 0F A4 3C F7 1B : F2
000330  85 DB 66 67 73 33 22 EC : E1
000338  D7 28 9D C7 B8 C4 77 DB : 31
000340  A1 5A D1 93 69 C2 75 03 : 02
000348  F6 70 E8 12 80 13 48 93 : CE
000350  CE 13 CC A7 F2 DC 09 5F : 8A
000358  E7 B8 12 D4 FF 52 FD 95 : 68
000360  DA E4 1E E4 A8 1E EE 93 : 07
000368  B0 22 9C F9 CE 7C A7 38 : 90
000370  E9 D7 18 A9 D6 1E 13 9E : 26
000378  FA 75 47 75 3A 83 B8 98 : 38
-----------------------------------
CKSUM:  CC 11 22 05 75 83 EF 74 719A

000380  63 11 22 0E D2 74 C7 5D : 0E
000388  30 86 1A 74 86 12 74 47 : 97
000390  41 39 E3 05 39 C3 96 98 : 8C
000398  22 F9 39 82 F5 30 07 1D : 1F
0003A0  2F C7 15 2F 85 DA 72 87 : 92
0003A8  09 39 23 7E 97 82 E1 38 : 15
0003B0  C2 DD 38 83 7A 97 62 12 : DF
0003B8  70 C4 14 E0 8B 44 BA 16 : C7
0003C0  69 70 2C 92 DC 40 4D F0 : F0
0003C8  AF 48 43 6C 90 43 F4 DD : 4A
0003D0  8D A2 59 8A B4 B2 1B 14 : A7
0003D8  DC 8A A4 AE 15 28 FC 54 : 45
0003E0  26 CC 3E 4D 90 A6 4D 78 : 78
0003E8  7A 95 02 95 29 C3 C4 A5 : FB
0003F0  0B 25 20 A3 4A 11 44 8E : 20
0003F8  05 0A 4F 07 49 A5 0E 51 : B2
-----------------------------------
CKSUM:  91 DE F7 DB 28 2C 02 71 49F7

000400  B0 70 9A 11 3E 9A 00 DD : 80
000408  25 C6 A1 33 67 2D 04 F2 : 49
000410  65 C6 9D 19 9C E5 4E 59 : 09
000418  1C 81 A7 11 4E 7C E7 3E : 44
000420  53 9C 73 9C 51 3C 3C 23 : EA
000428  50 3B E7 3D D0 DC 77 04 : D6
000430  F8 C4 0E 07 68 39 1D 70 : FF
000438  E8 61 8A 11 84 28 87 40 : 57
000440  51 8C 10 B0 E5 87 82 F8 : 83
000448  52 8B D0 F4 71 C5 30 E2 : E9
000450  87 C2 EC 54 0E 10 A9 1B : 6B
000458  F1 54 2E 06 C4 5B 8A B1 : D3
000460  BD 1B 41 08 3F 10 46 D8 : 8E
000468  5A 0A F1 66 20 0B 23 FA : 03
000470  01 E9 AF 28 76 C5 4B F3 : 3A
000478  A1 DA 1D 4D 59 D8 EC 4E : 50
-----------------------------------
CKSUM:  AD 8E 69 40 F2 10 15 F6 6B32

000480  E6 A8 B0 6A 4B 26 A0 B4 : 6D
000488  5F 16 CD 31 70 BD 2E 9A : 68
000490  52 F1 78 5F 2B 10 0D 19 : 7B
000498  04 D1 10 8D 09 0C BA 22 : 63
0004A0  17 24 52 E0 8C 67 C8 E5 : 0D
0004A8  B9 20 E4 12 7C 24 AB 8B : A5
0004B0  20 C3 E5 90 71 8B 91 00 : E5
0004B8  63 C0 9C AC 88 33 97 30 : ED
0004C0  95 A4 88 63 2E 61 EE 80 : 21
0004C8  C4 5D 09 89 AD 8C 55 D2 : 13
0004D0  98 BB 3C 63 2B 83 1B 68 : 23
0004D8  4C 75 A4 31 F6 94 C8 5A : 42
0004E0  73 23 6A 0C 95 D7 99 3B : 4C
0004E8  B2 32 97 66 65 6B F3 2D : D1
0004F0  6B 8C BD DC 99 8B 64 66 : 7E
0004F8  6D 99 9A BB B3 37 49 F4 : 82
-----------------------------------
CKSUM:  28 F2 85 3E 32 50 8F FF 6234

000500  CE 9A F3 3C 60 19 F3 64 : 67
000508  68 06 CC D0 4D A1 A1 18 : B1
000510  26 86 61 1A 21 DE 9A 29 : E9
000518  B7 34 63 70 68 E7 7E 69 : F4
000520  07 84 69 26 EC D2 8F 14 : 7B
000528  D2 CF 1C D3 0D E9 A6 9B : C7
000530  E3 4E 3C B3 4F 38 26 A0 : 6D
000538  7A 06 A2 70 8D 48 E1 9A : E2
000540  99 EB 9A A1 ED 1A A9 C4 : 33
000548  35 63 DC 35 73 DD 35 83 : B1
000550  DF 91 59 B8 A6 B6 71 E4 : 32
000558  57 2F 9F 22 BD 50 1C B2 : 22
000560  39 CA 9C B3 39 CB 9C B4 : A6
000568  4E BA 66 D3 AC 92 E7 3A : A0
000570  04 EA A6 85 3A 88 D9 30 : E4
000578  D3 4A 91 12 79 3A 68 E1 : BC
-----------------------------------
CKSUM:  AB C7 8D 7F 66 D6 17 D3 5F4B

000580  30 92 85 3A 49 48 9D 14 : C3
000588  A5 4E 7A 53 A7 39 2A 13 : DD
000590  05 35 E9 CC 4D 92 60 26 : 54
000598  CD 2F D1 FA 5F 25 72 72 : 2F
0005A0  93 72 9C 94 B2 4B C4 B3 : A9
0005A8  4E 32 6E D3 88 90 52 ED : 18
0005B0  0D 80 A6 F9 38 29 6E 97 : 92
0005B8  49 70 7F 5D 1E 9E 09 43 : 9D
0005C0  C3 2A 6E CE 87 88 75 3C : E9
0005C8  63 B1 BC 3B 9E 49 60 F2 : 44
0005D0  8B 26 F8 B4 6F CB 67 00 : FE
0005D8  B8 79 85 D3 82 5E 3C E2 : 87
0005E0  F9 E7 90 0F 44 82 7A 44 : 03
0005E8  23 84 43 3D 32 21 88 45 : 47
0005F0  38 64 63 D4 23 9E A9 20 : 5D
0005F8  E4 12 73 64 AA D2 41 85 : 0F
-----------------------------------
CKSUM:  7F 33 38 24 85 E7 8A 77 593F

000600  9C 83 8B 24 FA 13 D9 FC : B0
000608  1B 66 04 C9 A2 3F 70 4E : ED
000610  9E 6D 02 9F EB F8 52 50 : 31
000618  A5 70 1C 29 04 E8 74 C1 : 7B
000620  B0 D1 1F 13 52 A3 29 21 : F2
000628  6C A0 9E 52 0A B0 DA 46 : D6
000630  56 1A 96 A5 7C E9 5F 9F : 0E
000638  19 C1 2D 21 7C EA B0 F4 : 32
000640  70 DF 1F DA 78 CD 23 29 : D9
000648  F0 C8 5E D4 66 83 36 1E : 27
000650  EA 41 D1 C3 DD C6 F9 DC : 37
000658  70 61 ED 1C 68 E1 C7 5F : 49
000660  33 1F 7F 9A 20 76 4C 6F : BC
000668  76 8E 39 B0 FC 4E 30 A3 : 0A
000670  6E E3 16 0F A4 BF E6 4B : 0A
000678  FF 38 56 B0 C1 70 D2 35 : 75
-----------------------------------
CKSUM:  55 23 8C 76 83 42 6E 69 5ECC

000680  54 6E 6C 3E FC 6C 28 D6 : D2
000688  45 C2 AF 39 4B 01 30 7F : EA
000690  91 9F 6A 3E FE 58 38 56 : BC
000698  1B B7 1F 9D C7 CE E3 F4 : FA
0006A0  C1 AA 8D ED DA AB 83 21 : 0E
0006A8  FB ED 47 DF B4 0D D4 83 : 26
0006B0  9D 70 F0 AB 7D 0F 47 D2 : 4D
0006B8  69 E8 13 65 BB 2F 21 BA : 8E
0006C0  C6 5C 30 C1 CE B8 F6 E0 : 6F
0006C8  E7 5B B4 F4 29 01 4A F3 : 51
0006D0  BD 97 29 F2 16 CB 1D 7B : E8
0006D8  72 7F B5 53 E1 53 E0 8F : 9C
0006E0  06 73 FE B7 DB 19 66 74 : FC
0006E8  3A 63 15 74 8A 7B F1 D6 : F2
0006F0  CC 47 1C E6 BD 6D D6 22 : 37
0006F8  1E 7C A9 61 5B 1B BF F6 : CF
-----------------------------------
CKSUM:  0D DB 15 9A 3D 7C 5B 0E 534F

000700  B5 9C 88 FB 30 42 73 11 : CA
000708  DB 3F 7A D9 6F 5C 81 64 : 1D
000710  5E 2D 63 94 FD 8B 55 F5 : 54
000718  AD C8 FB 2C 65 8E 4A D5 : AE
000720  44 9E 4A 65 BC 4B FD 13 : A8
000728  0B 4B D7 85 26 F4 41 93 : A0
000730  59 6B 13 1F 9D B9 59 CE : 73
000738  88 69 9B 8F C0 B3 AE DF : 1B
000740  C6 B3 AD 23 1D 2E 71 3E : 43
000748  35 9C D0 66 64 7A 55 95 : CF
000750  D8 5C D3 F6 57 C6 6B 27 : AC
000758  4F DA 5F BD 4B E0 CB AE : E9
000760  7F 80 DD A8 65 5B 32 B7 : 2D
000768  8A DA 25 0F F9 11 01 F1 : 94
000770  B0 D9 A1 F7 12 BE 23 9E : B2
000778  DC 4C 8E 9F EB FA 31 2D : 98
-----------------------------------
CKSUM:  82 91 0F B5 BE D4 5B AD 8852

000780  B1 2D BF 8C 49 DB A3 E9 : D9
000788  ED 18 F6 EF 12 D9 D4 8D : 36
000790  5D 8A ED 62 35 63 DE 9F : 4B
000798  A6 97 C0 92 4F 77 40 00 : 95
0007A0  00 00 00 00 00 00 00 00 : 00
0007A8  00 00 00 00 00 00 00 00 : 00
0007B0  00 00 00 00 00 00 00 00 : 00
0007B8  00 00 00 00 00 00 00 00 : 00
0007C0  00 00 00 00 00 00 00 00 : 00
0007C8  00 00 00 00 00 00 00 00 : 00
0007D0  00 00 00 00 00 00 00 00 : 00
0007D8  00 00 00 00 00 00 00 00 : 00
0007E0  00 00 00 00 00 00 00 00 : 00
0007E8  00 00 00 00 00 00 00 00 : 00
0007F0  00 00 00 00 00 00 00 00 : 00
0007F8  00 00 00 00 00 00 00 00 : 00
-----------------------------------
CKSUM:  A1 66 62 6F DF 8E 95 15 2F3F
```

NEW PRODUCTS
新製品紹介

XVI環境の救世主となるか?

X-SIMM VI

Taki Yasushi 瀧 康史

湯水のようにあればうれしいメモリ。

X68030では安くなったものの，XVIメモリの定価は見直しもなく高いままだった。そのうちX-SIMM10の発売でスロットメモリが安く増設できるようになった。

XVIの場合，スロットメモリで増設すると実に10MHzのX68000と同程度以下のシステムポテンシャルしか引き出すことができない。これは拡張スロットを従来機と完全互換にするための処置。当初から中途半端といわれつつも，メモリ専用スロットで8Mバイトまで増設できたXVI。フル実装するには結局1スロット消費するまぬけなマシンとなってしまった。

安いメモリが出回るなか，XVIの内蔵メモリは安いところでも1Mバイト2万円。大昔に出た，純正の高いヤツしかないからである。8Mバイトにするためには，約12万も払わなくてはいけない。さらに4Mバイトを遅いRAMで増設したら，プラス4万円。しめて16万円……。

こんな最悪環境から，XVIにX-SIMM10を差す人も出てきた。悲しいことにXVIにX-SIMM10なんて差したら，SUPERと同等のマシンになってしまう。たとえ24MHzバリバリにクロックアップしたXVIでも，スロットメモリ上でプログラムを動かしたら10MHz相当だ。

デバイスドライバや常駐ソフトは普通，メモリの前から確保するって知ってるかな?　メモリが増えたと，1Mバイトディスクキャッシュなんて確保したら，これだけで速いRAMは塵となってしまう。どうせなら，IOCS系，FLOAT系，PCM8などのプログラムを速い最初の2Mバイトに常駐したい。

VRAMやI/Oが速いままなので，まったく10MHz相当とまではいかないが，スロットメモリ上では10MHzマシンとほとんど変わらないスピードでしか動かない。

ほら，なにがなんでも，速いRAMがほしくなっただろう?　でも高い。そこでX-SIMM VIの登場である。

▼
コストパフォーマンス
▲

この商品をまず簡単に説明しよう。

ターゲットはXVIとCompactXVIだ。ただ，この2機種ではメイン基板のメモリ専用スロットの形状が異なるため，基板は同じようでも別の商品となるのでご注意。

X-SIMM10同様SIMMは付属しない。自分で買うなり，販売店で揃えてもらうなりする。また，SIMMは72pinSIMMになっている。多分スペースの都合だろう。

72pinSIMMは3種類から選択できる。2Mバイト，4Mバイト，8Mバイトの3種。SIMMスロットは1スロットなので，2Mバイトや4MバイトのSIMMを使用すると，速い内蔵RAMとしてはその分しか増設できない。XVIのメモリ専用スロットには，8Mバイトまでのアドレス線しか出ていないため，8MバイトSIMMを載せても，内蔵2+8=10Mバイトという甘い計算どおりにはならない。あくまでも8Mバイト。余った2Mバイトは本当に無駄にしかならない。これはXVIの設計上，どうしようもないことだ。

さて，定価から考えてX-SIMM VIは実売1万5千円弱くらいになるだろうから，8Mバイト60ns，72pinSIMM(3万5千円)とあわせた値段は，だいたい5万ぐらいになる。純正で揃えたときより確かに安い。

2Mバイトユーザーならもう「買い」だろう。「最初は2MバイトのSIMMを……」などと考えているとバージョンアップ時に必ず損するので，いきなり8Mバイトをつけよう。

すでに親ガメ(2Mバイト増設)をつけていた場合はどうだろうか?　私が思うにXVIユーザーでいちばん多いのはこのタイプだろう。親ガメがいくらで売れるかにかかってくるが，全体が5万円なら差し引きして，考えてみるとよいだろう。私の計算だとすでに6Mバイトユーザーでも，X-SIMM VIで増設したほうが安そうである。

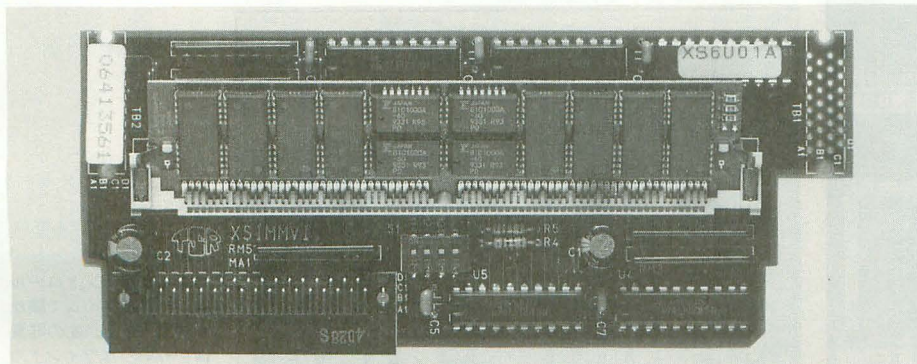
ただ，X-SIMM VIは純正品と違いROMソケットがない。ROMソケットはなくても困らないかもしれないが，Compactユーザーで数値演算プロセッサが使えないというのは痛いかもしれない。

▼
メモリの性能
▲

ここまででわかったとおり，コストパフォーマンスは悪くない商品だ。しかし，私がこれを最大に評価したいのは，60nsの高速なRAM(SIMM)が使えるということだ。

純正RAMは70ns。24MHzは完全なオーバースペック。拡張RAMはメイン基板上のRAMと違い，バスバッファ(AS244)が1段入っている。そのため，本体基板実装の2Mバイトは24MHzでバリバリ動いていても，拡張した6Mバイトでエラーが出るというケースも珍しくない。こういうのは普通のRAMチェッカでは出ないのがオツ。特に夏場。たぶん2台に1台の拡張RAMでエラーが出ている。

60nsが使えれば24MHzでも問題はない。夏場でも安心して使えるし，エアコンを切った夏場の部屋のなかでも動いている。

不安定なメモリほど心配なものはない。24MHzでどうも動作が不審で不安な方，考えてみてはいかがだろうか?

X-SIMM Ⅵ　16,800円　東京システムリサーチ　☎0425(28)1824

新製品

Mu-1 GS

Taki Yasushi 瀧 康史

X68000で音楽を楽しんでいるユーザーのみなさん，お待たせしました。発売が延び延びになっていた「Mu-1 GS」がとうとうお目見えです。ではさっそく，細かいバージョンアップ点を見てみましょう。

数少ないX680x0のアプリケーションのなかでも秀作である「Mu-1 Super」が，新たにGS音源に対応してリリースされました。その名も「Mu-1 GS」です。

そして単にGS音源対応だけではなく，操作などの細かい部分の見直しや，ステップエディタ機能の充実が図られました。したがって，「GS」と謳われてはいても，通常の機能で利用する限りは，MT-32などでも使用することができます。つまりMu-1 Superから音色エディタなどの特殊機能が排除され，機能アップしたということです。

GS専用として付加された機能といえば，レゾナンス，カットオフ，エンベロープ関連などのパラメータを，ボタンで簡単に処理できるようになったということです。これにより，直感的にはわからなかった音色効果処理の一環を視覚的に操作することができます。

ではここで，Mu-1シリーズを知らない人のために，簡単に説明してみましょう。

まずMu-1といえば，リアルタイム入力です。リアルタイム入力というのは，簡単にいえば，MIDI信号の録音です。MIDIキーボードなどを接続し，文字どおり，リアルタイムに入力(演奏)した曲データをそのまま録音するというものです。

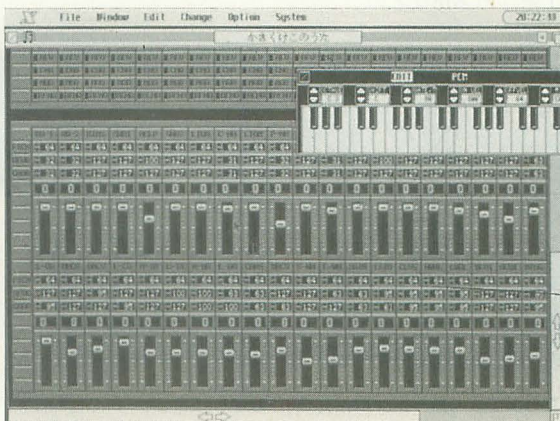
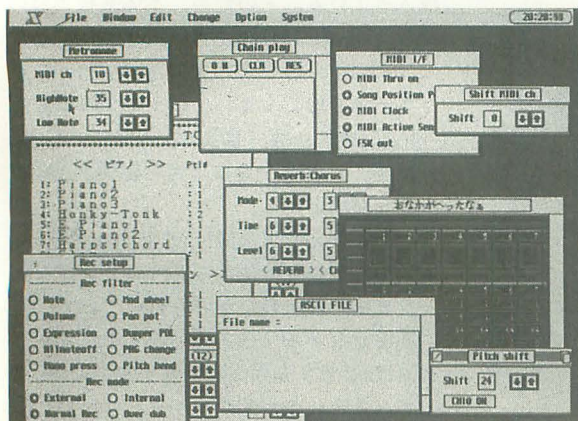
そのため，いかにも，機械で作ったというものではなく，人間らしい演奏データを収録することができます。

Mu-1はこの機能がなかなか卓越していて，4分音符240カウントと，とりあえず海外に出しても恥ずかしくはないぐらいの性能をもっています。確かに海外では，480カウントぐらいのリアルタイム入力ソフトがだいたいの標準ですが，日本国内のシーケンスソフトはたいてい48カウントぐらいで頑張っているのも事実です。Z-MUSICもデフォルトでは4分音符48カウントですから，240カウントでも十分使えるレベルだといえるでしょう。

エクスクルーシブやコントロールなど，ある程度フィルタリングしながら，録音することもできます。ある一定の部分だけ，録音時に演奏を間違えたときのために，パンチイン，パンチアウトという機能もあります。これは，その部分だけを録音する機能です。演奏途中でボリュームスライドバーを動かし，その部分の記憶もできます(しかしこれは1つのマウスではちょっと苦しいかも)。まさに「MIDI＝デジタル楽器のMTR」のノリを地でいっているといえるでしょう。

データ形式は，全世界で標準のスタンダードMIDIファイルを筆頭に，日本だけのローカル標準RCM(*.RCP)，MUSIC SX-68KなどのSCOファイル，OPMファイル，そして従来からのMu-1の内部データをそのまま吐き出した形のSNGファイルをそのまま読むことができます。このうち，Mu-1 GSからの新機能はRCM形式の対応でしょう。Mu-1 Superでは，この肝心なスタンダードMIDIのコンバータが，妙な変換をして，うまく使えませんでしたが，今回は完全に変換できるようです。

MTRと称される部分は，視覚的にいろいろなデータをリアルタイムに変えることができるものです。機能はまさにMTRのそれで，フェーダを利用したチャンネル個々のボリューム，パンポットなどの機能がついています。これはもちろん，再生中に設定することもできますし，ミキサー気分で，録音しながらフェーダを動かすこともできます。

また，コントロールパネルが貧弱なコンピュータ用音源では，個々の内部パラメータを簡単には操作できません。SC-55はまだいいものの，CM-300，CM-500はまったくこれらの機能がなく，演奏しながらちょっとレゾナンスをいじってみるなんてことができません。しかし，これらを簡単に操作するための操作パネルをMu-1 GSはもっているので，当然これもリアルタイムで録音することができます。これらの設定が

さまざまなコントロールは，リアルタイムで細かく調節できる。左の写真はいろいろなウィンドウを開いたところ。右はMTR機能だ

視覚的に操作できるというのも，なかなか便利な機能だと思われます。

■ ステップエディット ■

もともと，Mu-1は単なる演奏ツールです。それに，Mu-1SuperからMusicstudio Proを吸収して，リアルタイム入力ツールになりました。

先に説明したとおり，肝心のリアルタイム入力はわりと出来がよく，音楽を演奏するときの派手さも手伝って，なかなか秀逸なソフトに仕上がりました。

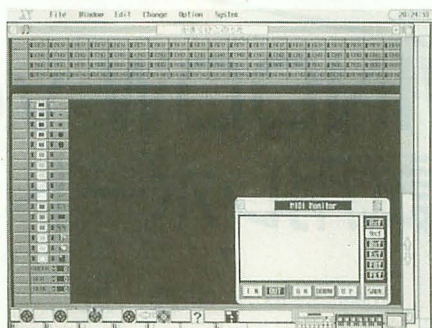
しかし，それはユーザーの演奏がある程度「上手ければ」という限定された条件のもとであり，失敗したときに，ある一部をステップエディットで直したいという場合には，融通が利かない一面もありました。

Mu-1 GSでは，この反省を大いに活かして，ステップエディットの充実が図られています。

まず見直された点は，エディットモード内で，キーボードオンリーでも作業ができるようになったということです。以前は，何をやるにも必ずマウスを触らなくてはいけなかったのに，今回はトラック内の編集ならばキーボードだけでできます。ステップ入力にはMIDIキーボードが実質必需品となりかけていましたが，どうやらその部分も，ある程度解消されたようです。

また，置換などといったマクロ的な動作も改良されています。なかでも私が気に入った機能は，置換文字列にメタキャラクタのようなものが使えるということです。利用する機会はそれほどはないかもしれませんが，ある一定の音だけ置換するなどといった強力な置換機能は，その場その場でいろいろと役立つことでしょう。

さらに役に立つ改良は，ランダムにばらつきを与える機能です。これはどういったものかというと，ある範囲からある範囲までマークをし，その音をプラスマイナスnの範囲で，ばらつきを与えます。

Z-MUSICなどで，まるっきりステップエディットした音楽データは，音にメリハリがなく，機械的な雰囲気がするものです。MIDI楽器が高音質化し，音にリアリティが増してきたとしても，ベタで音を鳴らしてしまったのでは，気品も何もありません。

簡易グラフィックエディタもついている

実際に演奏している状態

そのため，最近のZ-MUSICデータはZコマンドを酷使し，音にばらつきを与えているようです。しかし，効果が大きい反面，猛烈に手間がかかるのも事実です。

しかし，Mu-1 GSでは，このランダム機能をうまく利用し，この作業をかなり楽にこなすことができます。まず最初にある程度ばらつきを与えます。これだけでも，データの雰囲気がぐっと変わってきます。そしてこれをスロー再生しながら聴き，妙なランダム変換をした部分を自分の手で入れ替えていくのです。

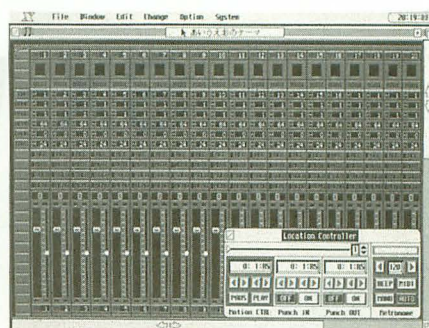
これだと，1音1音，ベロシティを考えながら打ち込むよりずっと楽です。

Z-MUSICのようなMMLではステップベロシティのばらつきを与えるだけで精いっぱいですが，Mu-1 GSでは，ステップカウントが小節線からの絶対値なので，音の鳴るタイミングにもばらつきを与えることができます。

今回の新製品の主な目的はGS対応という話でしたが，それだけではなく，こういった部分の改良もありがたいことです。

■ まとめ ■

欲をいえば，そろそろSX-WINDOW対応へと移行してほしいところです。ベースとなるミュージックシェルのバージョンアップも大変でしょうし，ユーザーも複数のウィンドウシステムを覚えることは大変ですからね。

しかし，細かいところにはまだ不満点はあるものの，このMu-1 GSは，すぐれたソフトであることは間違いありません。そろそろキーボードを買って，リアルタイム入力したい人，MMLでの表情つけに限界を感じている人，そういった人にぜひお勧めしたいツールです。

さて，ちょっと裏技ですが，Mu-1 GSのリアルタイムを悪用して，面白いことができます。必要機材は2台のX680x0とMIDIボード。

1台で，MIDI対応ゲームを走らせ，もう1台で録音。これで，「グラディウスII」などの，MIDIオンリーの素晴らしいデータがすっぱ抜けます(ただし，著作権の問題がありますので，こういうのを人にあげたりしちゃ駄目だよ。絶対に！)

こうすれば，このような出来のよいデータを参考にしたり，「出たな!! ツインビー」のSC-55用データのように，SC-55mkIIで不都合がある部分も，修正して再生することができます。

かくいう私も，いま，「グラディウスII」の音楽データを聴いているところです。スタンダードMIDIからZMDへのコンバータを利用して，SX上で原稿を書きながら聴いているというわけ。

考えてみると当たり前の話ですが，私もこのデータ引き抜きテクを聞いたときには，コロンブスの卵だなぁと思ったものです。Mu-1 GSは240カウントですし，コントロールなどもそのまま録音することができるので，こういうことが可能なんですよね。

X68000用　5"2HD版　28,000円(税別)
サンワード　☎044(855)4335

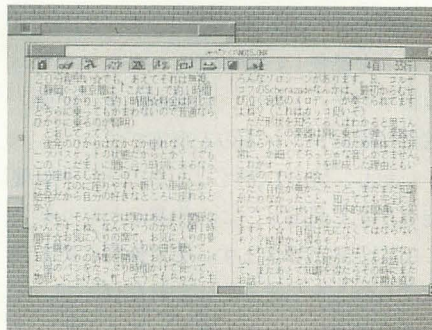
SX-WINDOW ver.3.1 試用レポート

Nakamori Akira 中森 章

SX-WINDOWが3.1にバージョンアップされました。今回のバージョンアップの目玉はシャーペン.Xの機能アップとコンソールウィンドウです。それでは，詳しく紹介していきましょう。

マニュアル

SX-WINDOW ver.3.0からは2冊のマニュアルが変更になりました。「導入マニュアル」と「日本語マルチフォントエディタユーザーズマニュアル」が，それぞれ「拡張マニュアル」と「シャーペンユーザーズマニュアル」に変更になっています。その他のマニュアルに変更はありません。これから見る限り今回のバージョンアップはほとんどシャーペン.Xのバージョンアップと思ってよいでしょう。

「拡張マニュアル」はインストールの方法とSX-WINDOW ver.3.1の特長と拡張部分の説明からなっています。これを読めば，今回なにがバージョンアップされたのかわかるようになるでしょう。「シャーペンユーザーズマニュアル」はシャーペン.Xのユーザーズマニュアルです。ver.3.0のものから全面改訂されました。ページ数は約50ページ増えて使用例も豊富になり，以前と比べると非常に読みやすいマニュアルになっています。加えて外部コマンドの説明と外部コマンドの作成方法まで書かれていてシャープのシャーペン.Xにかける意気込みが伝わってきます。聞いた話によると，この「シャーペンユーザーズマニュアル」はシャーペンによって制作されたのだそうです。これはもうDTPソフトといっていいかもしれませんね。

X68000用3.5/5″2HD版22,800円(税別)
シャープ ☎03(3260)1161

縦割り横割り自在のシャーペン

ディスクの構成

・Human68k ver.3.0システムディスク
・SX-WINDOW ver.3.1システムディスク
・辞書ディスク
・SX-WINDOW ver.3.1アプリケーションディスク1
・SX-WINDOW ver.3.1アプリケーションディスク2
・SX-WINDOW ver.3.1アプリケーションディスク3
・拡張ディスク

以上の7枚のディスクによって構成されています。今回のバージョンアップに関係する拡張機能（アプリケーション）のほとんどすべては拡張ディスクに集められています。

では，もう少し詳しく変更点を見ていきましょう。

●Human68k ver.3.0システムディスク

HUMAN.SYSのバージョンは3.02，COMMAND.Xのバージョンは3.00で変更はありません。

ASK68K.SYSのバージョンも3.01から3.02に変更になっています。これはいくつかのバグフィックスと思われますが，細かいことはわかりません。

あと，FLOAT2.XとFLOAT3.Xのバージョンが2.02から2.03へ，FLOAT4.Xのバージョンが1.00から1.02に変更になっています。これはバグフィックスというよりも，若干の処理の高速化です（恐ろしいことに自己書き換えをやっている，起動時に1回だけ行われるので問題はないと思うが）。

●SX-WINDOW ver.3.1システムディスク

FSX.XとSXWIN.Xのバージョンは，それぞれ，3.01から3.10，3.00から3.10に変更になっています。SXKERNEL.Xに変更はありません。マニュアル自身に変更がないので，この変更はバグフィックスあるいは高速化が目的だと思います。ただ，FSX.Xではシステムコールが17個増えているようです。

SHELLディレクトリ内のファイルでは，

BUILTIN.LB
ICON.LB
SXWIN.ENV
SXWIN.X
SYSTEM.LB
TITLE.X

が変更されています。システムの変更にかかわるファイルばかりなので当然といえば当然でしょう。TITLE.Xなどは画面に表示する絵のバージョン表示が3.0から3.1に変わっただけです。新たに追加されたのが，

SXCON.X

です。これはコンソールデバイスドライバです。スタートアップメンテに登録しておくことで，シャーペン.Xのコンソールモード（あとで説明します）などが使用可能になります。

●辞書ディスク

誰でも変更はないと思いがちですが，約60の単語がX68K.DICに新規登録されてい

ます。

新規の単語の大半はアゼルバイジャン，ウクライナ，ウズベキスタン，エストニア，カザフスタンなどの地名ですが，ナタデココとかブルセラ，ノルディック，サポーター，ワールドカップなど最近有名になった単語も含まれています。

●SX-WINDOW ver.3.1アプリケーション1，2，3

次のファイルが変更になっています。

GRW.X
コントロール.LB
コントロール.X
クリップボード.X
パターンエディタ.X
IFM.X用のフォント

コントロール.XではX68030で指定可能になったMPUのキャッシュのON/OFFの設定ができるようになっています。パターンエディタ.Xはエディットできるパターンサイズが，以前は最大128×128ドット(縦横の積が16384以下)だったのが，ほとんど無制限になっています。これ以外のファイルの変更点はよくわかりません。アプリケーションに変更があるときはその旨をマニュアルに記述してほしいものです（コントロール.Xは拡張マニュアルに説明がありますね)。ただ，IFM.X用のフォントはタイムスタンプを保存せずにコピーしただけなのではないでしょうか。

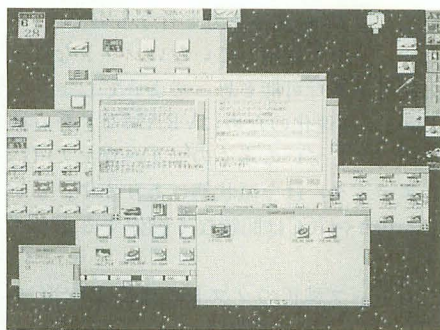
また，次のファイルが追加されています。

カタログ.PEN

これは，シャーペンの機能をなるべく多く使用して作成したSX-WINDOW ver.3.1の紹介文です。イメージデータやドローデータが埋め込まれていれば，よりシャーペン.Xの機能をアピールできたのではないかと思うのですが，たいした機能は使用してありません。昔のシャーペンでできないことはGScriptによる角の丸い色つき罫線枠くらいなので，もっと派手にやってもよかったのではないでしょうか。

●拡張ディスク

今回の拡張部分はすべてこのディスクに収められているといっても過言ではありません。このため，もっとも手を抜いたインストール方法は，SX-WINDOW ver.3.1システムのFSX.XとSXWIN.X（とSHELLディレクトリの内容）を入れ換えて，拡張ディスクの内容（と上記のアプリケーションディスクの新規ファイル）をアプリケーションを格納してあるディレクトリにコピーするだけです。

このディスクはインストーラ，高速化されたというフォントマネージャ（IFM.X, IFM.ENV, IFM.LB）とビデオマネージャ（IVM.X, IVM.LB），シャーペン一式，その他のおまけからなっています。

SX上からキャッシュが設定できるようになった

インストーラ

おまけのひとつめはシャーペンのマニュアル中で新たな外部コマンドを作成する例題として説明してある外部コマンドのソースファイルです。

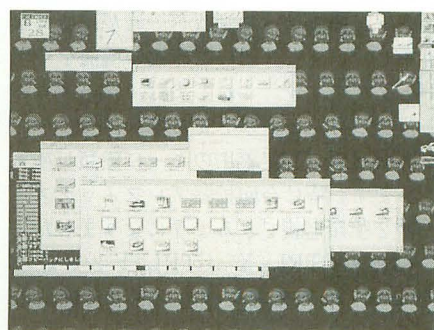
おまけの2つめはSXKEY.Xというコマンドです。これはSX-WINDOWのテキストエディットのキー入力処理を拡張するプログラムです。スタートアップメンテに登録しておくと，オブジェクトの選択時に，ダブルクリックでワード，トリプルクリックで行の選択ができるほか，シャーペンでのキーコントロールがそのまま使えるようになります。

おまけの3つめはシャーペン.Xの実際の外部コマンドであるRECT(GScriptの罫線）のソースファイルです。独自の外部コマンドを作成するときの参考になるでしょう。ただし，このファイルはGCCでなければコンパイルすることはできません。また，外部コマンド作成用に必要なライブラリが付属していています。

おまけの4つめはテキストエディットver.3.00以降のシステムコール（C言語版）のリファレンスとそのライブラリです。どうせなら，すべてのシステムコールのリファレンスをつけてほしかったところですね。

最後のおまけはシャーペン用ディザパターンのサンプルです。4種類のディザパターン定義用ファイルが付属しています。

シャーペン.Xの変更点

シャーペン.Xとはver.3.0のマニュアルのタイトルにもあるとおり，基本はマルチフォントテキストエディタです。今回，SX-WINDOWがver.3.1にバージョンアップされるにあたり，ワープロとして利用できるように文章の表現力を強化してあります。また，もうひとつの目玉はコンソールモードが使用できることです。

●ワープロ機能の強化

・インラインかな漢字変換をサポートしています。これは，シャーペン.Xとその付属モード（エディタ，タイプ，コンソール）のときだけ有効です。昔のエディタ.Xなどをアクティブにするとどこからともなく，従来の「かな漢字変換ウィンドウ」が現れてきます。

・網掛け，下線，上線，中線の種類が増えました。

・強制改行で改行の幅を変えられるので，大きな文字やドローデータの横や上に複数行の文章を並べることができます。早い話が，キャンバス.XやEasydrawから持ってきてペーストしたイメージデータやドローデータに文字を上書きできるわけです。

・文字にPSETやXORなどの描画モードが設定できます。これで背景にイメージデータなどがあるとき効果的に文字を貼り付けることができるでしょう。

・従来の罫線文字での罫線入力に加えて，GScriptでも罫線入力ができるようになったので，角を丸めたり色をつけたり太さを変えたりできるようになりました。楕円形の枠も描けます。罫線が自由に描けるようになってやっとワープロらしくなってきましたね。

・画面の拡大・縮小表示（25%～200%）ができるためレイアウトを確認できるようになっていますが印刷時とは独立して機能します。

コンソールもマルチタスク

・ページ枠表示，強制改ページが可能になりました。また，ページ印刷の機能をサポートしています。これでまた，ワープロに一歩近づいたというところでしょう。
・キーバインド定義用のキーに種類が増えています。特に今回拡張プレフィックスキーが2種類定義できるようになりました。このためCTRL-X CTRL-Fとかいう Emacsライクなキーバインドも可能になるでしょう。

●コンソールモード

シャーペン.XはHuman68k用のソフトをウィンドウ内で動作させるためのコンソールモードを装備しています。このようにHuman68kのCOMMAND.Xが動作しているウィンドウをコンソールウィンドウと呼びます。コンソールウィンドウは同時に複数個オープンすることができ，その場合，各ウィンドウはマルチタスクで動きます。

コンソールウィンドウは，通常は，シャーペンの外部コマンドで呼び出しますが，システムアイコンのポップアップメニューから選択することもできます。この点，コンソールウィンドウはシャーペン.Xの1機能としてではなく，SX-WINDOW ver.3.1に付随する特殊なウィンドウであると考えることもできます。

コンソールウィンドウの中では，Human68kと同じようにキーボードからのコマンド入力が可能です。入力されたコマンドはコンソールウィンドウの中で実行されます。この場合，実行可能なソフトは，グラフィック/スプライトを使用せず，VRAMを直接操作しないDOSコールを用いて文字表示/文字入力を行うものです。グラフィックを使用していないソフトならほとんどが実行可能と思ってよいでしょう。しかし，高速化のためにVRAMなどを直接アクセスしている多くのフリーウェア（特にエディタ）は全滅状態にあります。しかし，fishやGCCは問題なく動作するようです。

シャープ純正のソフトではX-BASICやED.Xがちゃんと動作します。GCCはエラーが発生するとエディタ（環境変数「満里奈」で指定するやつね）を呼び出す疑似統合化環境をサポートしていますが，「満里奈」にED.Xを指定しておけば，コンソールウィンドウの中で疑似統合化環境を使用することができます。これはちょっと感動ものでした。

スプライトやグラフィックが使用できないのでX-BASICが動作してもあまりうれしくありませんが，簡易計算機くらいには利用できるでしょう。とはいえ，過去に私が作ったプログラムではデータベースソフトや音楽プログラムがちゃんと動作しています。

コンソールの役割をしているとはいえ，これはSX-WINDOWのウィンドウのひとつにほかなりません。ウィンドウとしての機能も持っています。

まず，画面上に表示してある文字データをマウスでカット&ペーストできます。

ファイル名やディレクトリ名をキーボードなどから入力する代わりにファイルのアイコンをドラッグしてくるとカーソルの位置にそのファイル名が埋め込まれます。これもちょっと感動ものですね。

また，画面表示はバッファに蓄えられていきます。このため，画面表示から消えた過去の出力をスクロールして見ることができます。ちょっと大き目のサイズのテキストファイルをタイプしたとき，CTRL-Sで表示を止めながらちょっとずつ見る（こんなことをするのは私だけかな）という作業は不要になります。いったん全部をタイプしてから，あとでゆっくりとスクロールさせて見ればよいのです。

さらに，ファンクションキーなどの設定をコンソールウィンドウ（シャーペン.X）に対するものか，コンソールの中のHuman68kに対するものかを指定できるようになっています。XF3キーを押しながらキー入力をすると強制的にHuman68kへの入力になりますから，すべてのファンクションキーをコンソールウィンドウに対するものに設定しておいて，キー入力の対象をXF3キーで適宜切り替えることも可能です。

あと，めぼしい機能としては，
・カラー表示にできる
・フォントのサイズを変えることができる（12ドット，16ドット，24ドットから選択）
・1行の桁数を変えることができる（64文字，80文字，96文字から選択）

といったところでしょうか。

SX-WINDOW ver.3.1の感想

私は，コンソールさえあればとSX-WINDOWを使うたびに思っていました。フリーソフトでいくつかコンソールもありますが，ウィンドウのセンスが悪かったりスピードが遅かったりと，どれもしっくりしませんでした。そんなときに登場したシャーペン.Xのコンソールウィンドウは渡りに船といったところです。

あと，シャーペン.Xがどんどんワープロ指向になっているのもすごいと思います。シャーペン.Xとお絵描き用のEasydrawがあればもうワープロなんて不要になるんじゃないでしょうか（EGWordの立場は？）。お店でSX-WINDOW ver.3.1を手にしたとき，私は迷わずEasydrawを買うことにしました。私のほかにもSX-WINDOW ver.3.1とEasydrawをペアで買う人が多いようで，Easydrawを探すのに苦労しました。

そこで感想。シャーペンとEasydraw。これさえあればあとはいりません（ウソ）。

SX-WINDOWのインストール

コンソールとインライン変換のおかげでSX-WINDOW ver.3.1は日常的に使えるウィンドウ環境となってきました。この環境を享受するには新しいシステムをインストールしなければなりません。

拡張ディスクにはちゃんとインストーラがついているのですが，はっきりいってあまり使いやすくはありません。

複数の項目について一括処理を行うのに，転送先指定は各項目で個別に行わなければならないのはいただけません。画面が綺麗にまとまっている半面，必要な情報が一覧できないので確認作業が煩雑になります。ボタンひとつでインストールという次元にはまだまだ遠い道のりがありそうです。幸い，SX-WINDOW ver.3.1のシステムは奇怪な圧縮を施されてはいないので，普通の人は手作業でインストールしたほうが安全でしょう。

余談ですが，シャーペンのプリンタ出力時なども，なにが設定されているかわからないので，印刷前には用紙設定や印刷時設定/倍率などをいちいち確認しなければなりません。どうも，こういったユーザーインタフェイスのノウハウが足りないようです。

さて，とにかくこれまでの環境がなくなってしまうと環境整備にやたら手間がかかってしまいます。

特になくなると困るのはアイコンとポップアップメニューの設定などです。アイコンはICUP.X（1994年3月号付録ディスクに収録）を使って対処するとして，問題はメニューです。これはBUILTIN.LBの内部にリソースとして保存されています。SX-WINDOW開発キットのリソースエディタを使ってそれらの設定部分を取り出し，組み換えてやれば元の設定のままでシステムをバージョンアップできます（ARLK.X，RSC.Xでも可）。

これらの設定はDiME，ShMEというタイプのリソースになっていますので，従来使っていたシステムのBUILTIN.LBからそれらをすべて抽出し，新しいBUILTIN.LBに加えてみてください。

アクセラレータを作る（その5）

ついに動いたアクセラレータ

Ishigami Tatsuya 石上達也

トラブル続きで難航していたアクセラレータ製作ですが，試験基板をいくつか作るうちに，ついに動き始めました。現在，CPU68EC030の20MHz＋68882という構成です。ここまでの経緯と現状を解説します。

早いもので，前回の「アクセラレータを作る（その4）」からすでに1年がたってしまいました。

思い起こせば，この1年の間にいろいろなことがありました。で，いろいろなことがあった末にとうとうアクセラレータが動いたのです。

MACHについて

前回までの記事でアクセラレータで用いるPLD（Programmable Logic Device）にはLattice社のGAL（General Array Logic）を使用していました。

PLDとは手軽に作れるカスタムICのことですが，いくら手軽に作れるからといっても，やはり専用の工具は必要になります。

この工具と材料の価格，入手しやすさなどを考えて，PLDのなかでもGALと呼ばれるICを採用する予定でした。

しかし，実際に回路を組み上げてみると，このGALではやや機能が足りない場面があり，今回からはPLDとしてAMD社のMACH-210というICを使用することにしました。MACHとはMacro Array CMOS High-densityの略だそうです。

図1-1～1-3にMACH-210のピン配置図，ブロック図，PAL部分のブロック図を掲載します。

GALではなにが役不足だったかというと，出力段階のフリップフロップ回路の非同期セット機能です。

普通，GALといえば，そのシリーズのなかでもGAL16V8かGAL20V8を指すのですが，これらのICでは出力信号をフリップフロップに通した場合（Register Mode），その出力信号の非同期セット/リセットが効きません。

出力信号の非同期セット/リセットを行うにはGAL18V10かGAL22V10を使わなくてはなりません。仮にこれらのICを使ったとしても，非同期動作できるのはリセットだけで，セットは同期動作になってしまいます（フリップフロップと出力端子の間にインバータをつけることができるのでセットを非同期動作させることもできますが，この場合，リセットスイッチが同期動作になってしまいます）。

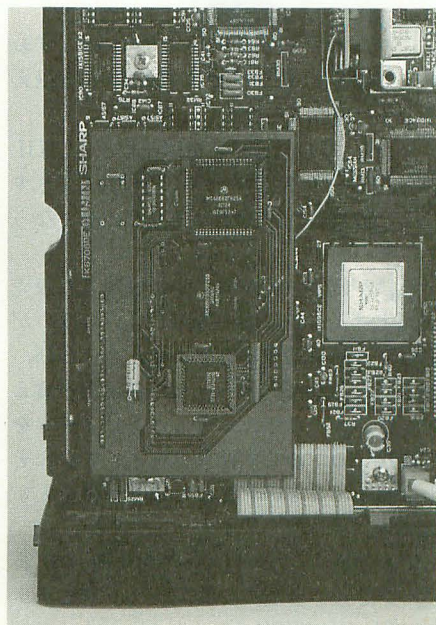



写真1　このように装着される

回路図を見ればわかるのですが，アクセラレータにはいまのところ，リセット動作を行っているところはありません（X68000のインタラプトとかリセットではなく，MACH-210の出力リセット）。

ところが，逆にセットを見るとXRW以外のすべてのフリップフロップ部分が使っています。そのうちセット信号を共有しているのは一対だけです。GALはひとつのパッケージ内ですべてのセット/リセット信号を共有しなければならないので，ここでは不適切です。

今回の回路まででしたらなんとか工夫を凝らしてGALで実現できるかもしれませんが，68030をX68000のクロックと非同期に動作させる場合，結局この場所はいろいろ変更を加えなければならない場所でしょうから，いまのうちからMACHに置き換えておくことにしました。

広大なIBM-PCの市場

今や世界で，出荷台数1千万台とも2千万台ともいわれるIBM互換機ですが，アクセラレータを作成していて，つくづく，その規模の大きさに驚かされました。

たとえば，今回のプリント基板作成に使用したEASY-PCというソフトウェアは，「Byte」という雑誌のThe Buyer's Martというコーナーで見つけました。このコーナーはA4判の1ページを18分割し，その枠ごとに1社ずつ広告を載せているというものです。1社当たりの広告スペースがOh!Xの編集後記ひとり分よりも小さいのです（ユニバーサルプログラマのほうは9ツ切広告，編集後記とほぼ同じ大きさ）。

さすがに，いきなり注文するのは恐かったので，冷やかし半分で資料を請求してみました。

資料が到着して，びっくり。デモ版，簡易マニュアル，そして，豊富なユーザーサポート（私はクイーンズ・イングリッシュを話せないので半分関係ないけど）。

とても，12行の広告ですまされるような製品には思えません。国内だったら，カラー見開きで広告してもおかしくないくらい，しっかりとしたものです。

昔，X68000が10万台売れたという話をどこかで聞いたような気がしますが，この編集後記よりも小さい広告しか出していないCADソフトは，2万コピー近く売れているそうです。しかも，雑誌の広告からはわかりませんでしたが，DMを見て，デジタル回路シミュレータ，アナログ回路シミュレータなど，ほかにもさまざまなシリーズ商品を揃えていることがわかりました。

ちなみに，CADよりも市場が狭いと思われるユニバーサルプログラマですが，今回使用したものは，前機種の6万人に及ぶユーザーの声を参考に改良を加えたものなのだそうで……。

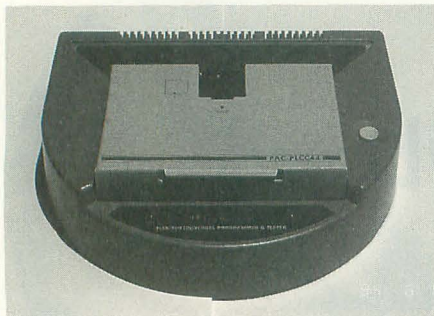



写真2　今回使ったユニバーサルプログラマ

MACHを扱えるプログラマ

とりあえずX68000はどこかにおいておいて，最近はPC-9801でもGALなどを扱えるようになりました。

しかし，MACHだとかPEELだとかXILINXだとか少し凝ったPLDを扱おうとすると一般に百数十万円の投資が必要となってしまいます。

値段が値段ですから，同じ機械でも販売代理店によっては数十万円くらいの違いが表れても不思議ではありません。今日び，20万円もあればちょっとした互換機が買えてしまいますので，ホストコンピュータを国産機にこだわらず個人ベースでもなんとか買えそうなものを探したほうが得策といえます。

とりあえず，「トランジスタ技術」の広告ページをめくります。㈱エルミックシステムのEL-WRITER Model All-03A。「4000種ものデバイスに対応，150種のアダプタを用意」。標準価格218,000円（＋アダプタ代）。

で，パラパラッとページをめくって，㈱バリオンのAll-03A。商品写真をよーく見ると，左下のほうにHi&Lo Systemsと書いてあります。先ほどのものとまったく同じもののようです。本体価格98,000円（＋アダプタ代）。

私の知人にこれを購入した人がいますが，サポート体制もしっかりしていて安心だそうです。

さて，目を海外に移して月刊「Byte」。Tribal Microsystems IncのTUP-300。今度は1ページを9つに区分けた広告で発見。いくら目を凝らしても，この広告からはどのような商品なのかわからないので，商品資料を請求しました。すると，ケースの形は似ているのですが，ロゴはしっかりと「Tribal Microsystems Inc」と書いてあります。795ドル。私は結局ここで買ったのですが，中身はAll-03Aと変わりありませんでした。書き込みプログラムの中にも，コピーライトが「Hi-Lo Systems」のままのところがありました。

ここは，クレジットカードによる送金が効かなかったため，送料のほかにも送金手数料やなんやかんやで，けっこうかかるのですが，アダプタが国内に比べ半額～1/3ですので，まとめ買いをする方にはおすすめです。

さらに，ワールドワイドに攻めて「電子情報」を見てみましょう。さすがに台湾の「トランジスタ技術」といわれるだけあって，2～3カ月前に「トランジスタ技術」誌上で見かけた記事やイラストがふんだんに使われています。

そのなかにHi-Lo Systemsの広告がありました。台湾で使われている中国語が得意な人はいきなり産地直送を攻めてみるのも面白いかもしれません（英語で問い合わせたところ，返事がもらえず具体的な販売価格がわかりませんでした。ちゃんちゃん）。

回路について

図2-1に全体の回路，図2-2，2-3にMACHの等価回路を示します。

理論的には図2-2，2-3に相当するような回路を74シリーズのICで組めばまったく同じ動作をするはずですが，実際はなかなか難しいかもしれません。

図3-2は図3-1の回路におけるタイミング図です。入力された信号とその反転信号を比較し，一致していたら「1」，不一致なら

図1-1　MACH-210ピン配置図

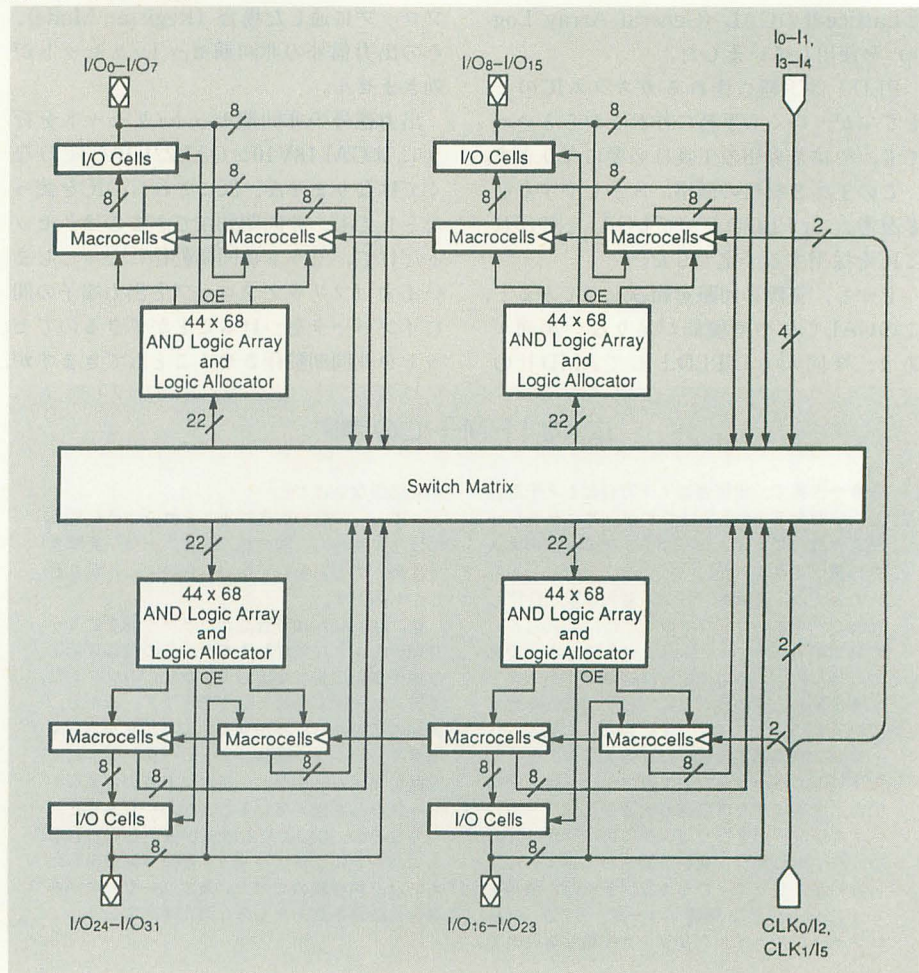
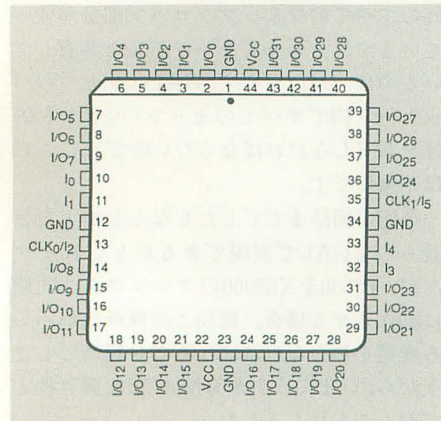



図1-2　ブロック図

ば「0」という回路です。そんなもの一致するわけはありませんので，常に出力は「0」のはずです。

しかし，実際にはインバータにおける遅延時間があります。この瞬間，インバータの出力は不安定で期待どおりの動作をしているとは限りません。この瞬間に，たまたまEX-ORゲートが機敏に動作したりすると，その瞬間の出力は「1」になります。

ほとんどのPLDの場合，入力信号は反転しようがしまいが結果に関係なく一様に遅れますので，そのような心配はありません。

実際に，読者の方からもPLDを使わずに同様の回路を動かしたという方もいらっしゃいましたので止めはしませんが，勇気と忍耐を持って覚悟のうえで挑戦してください。

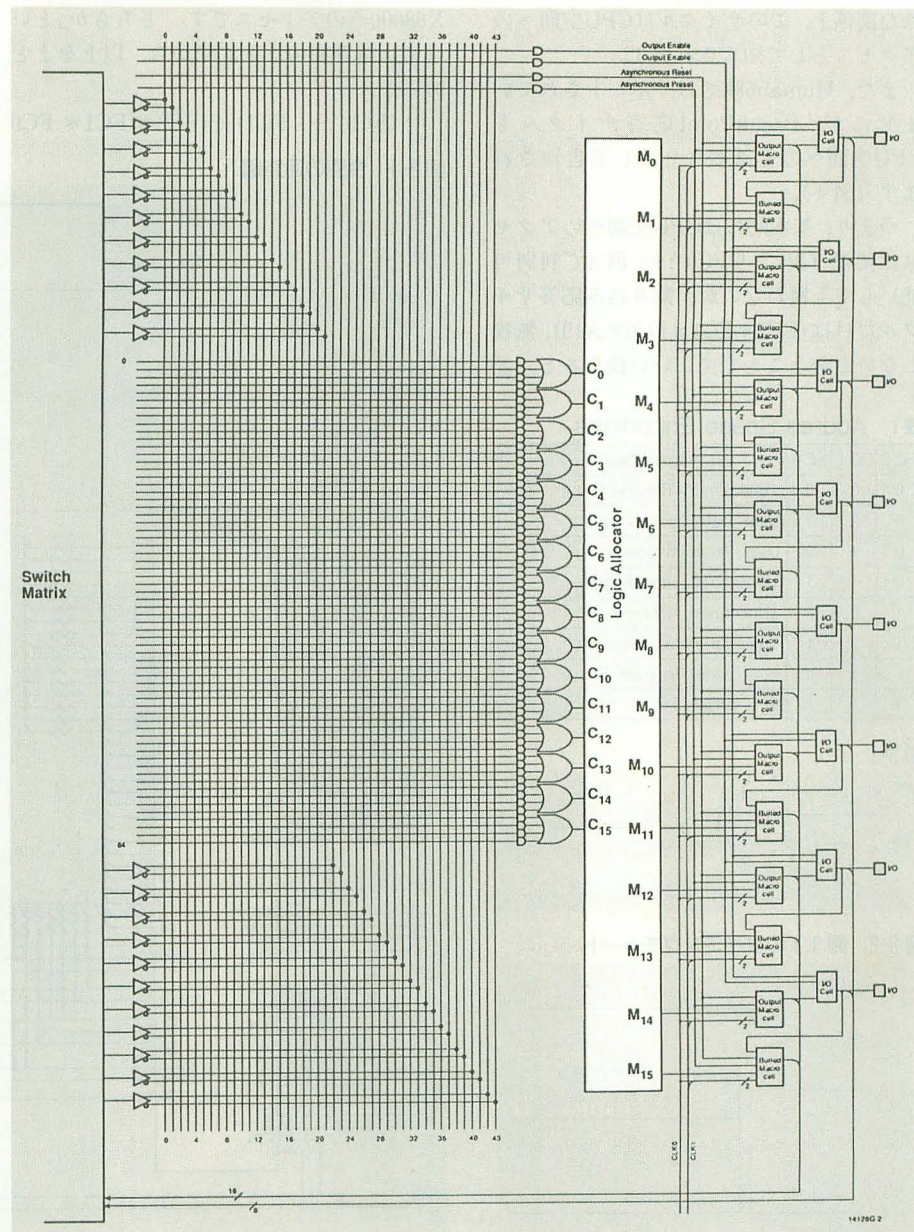
リスト1について

リスト1は結局，図2-2，2-3と同じことをMACHにやらせるためのデータにほかならないのですが，せっかくPLDを使って回路図を離れて設計したので，リスト1に従ってMACHの働きを説明します。

CONTEN:

68030は68000に比べ扱うメモリ空間の種類が拡張されています。68000はスーパバイザ空間(データ＋コード)，ユーザーモード空間（データ＋コード）の4種類だけでしたが，68030では「CPU空間」がサポートされています(表11)。CPU空間とは，CPUが主にコプロセッサとの通信に用いるためのメモリ空間で，通常のメモリ（メインメモリ，グラフィックメモリなど）は，このタイミングではアクセスできません。

ただし，例外があり「割り込み応答サイ

図1-3　PAL部分のブロック図

PALASMのバグ?

UDS := (RW + $\overline{A0}$) * CONTEN

と，

LDS := (RW + A0 + $\overline{SIZ0}$ + SIZ1) * CONTEN

一見どちらも組み合わせが違うだけで，式の形にまでは違いないように見えます。しかし，PALASMの中では，違う解釈のされ方をしているようです。

MACHは，4項までを論理和結合し1クラスタとし，クラスタを論理式結合して，ロジックを組み立てます。/UDSを展開する（括弧を外す）と，

UDS := RW*CONTEN
 + $\overline{A0}$*CONTEN

で，2クラスタ使用します。同様に，$\overline{LDS}$を展開すると，

LDS := RW * CONTEN
 + A0 * CONTEN
 + $\overline{SIZ0}$*CONTEN
 + SIZ1 *CONTEN

となり，4クラスタも使用します。ところが，ド・モルガンの定理（$\overline{(A*B)}=\overline{A}+\overline{B}$）を使って，

$\overline{LDS}$:= ($\overline{RW}$*$\overline{A0}$*SIZ0*$\overline{SIZ1}$)
 + $\overline{CONTEN}$

のように変形すると，2クラスタですむわけです。同じ機能を実現するのに，少ないクラスタ数ですめば，クラスタに限りのあるICの集積度をより上げられるわけで，とてもよいことです。

あまりにも基本的なことなので，コンパイラのオプションスイッチをすべて切っても，この最適化は行われているようです。

で，ここまではまったく問題ないのですが，リスト1では，この項に関して，

LDS.SETF＝/DS
LDS.RSTF＝GND

と記述された部分があります。

$\overline{LDS}$:=～

を，

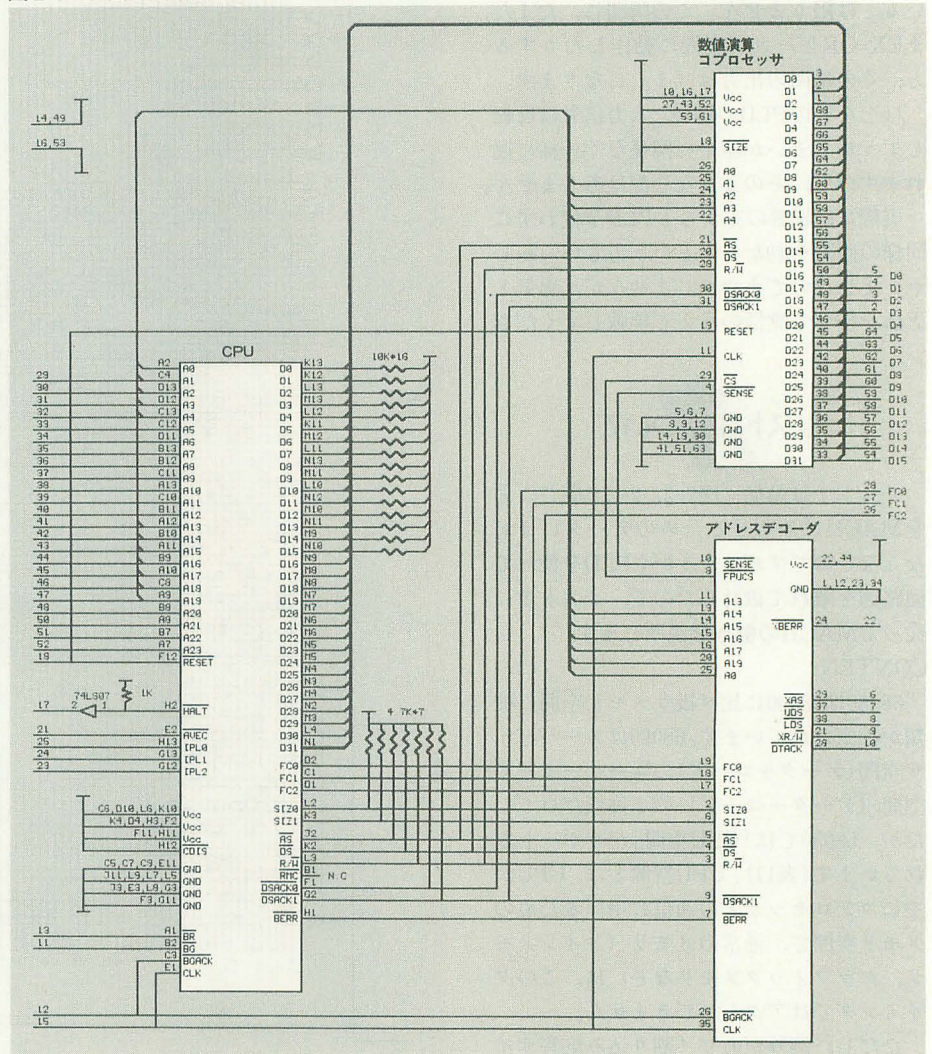
LDS :=$\overline{～}$

と書き換えたからには，この部分もひっくり返されるべきだと思うのですが，なぜか，コンパイラはやってくれていません。写真1のような回路でチェックしても，LDSの動作は不安定でした（本来，コンパイラにはシミュレータというものがついてくるので，コンピュータ上で検討できるものなのですが，今回は事情が事情なので，やむなくチェック回路を実際に組み立てました）。

以上のような理由で，リスト1のUDS，LDS部分は，ややトリッキーな記述になっています。PALASM以外のコンパイラを使用する方は，等価回路を参考に通常の方法で記述してください。

クル」を表す$\overline{\text{IACK}}$端子が68030では廃止された関係上，このサイクルはCPU空間へのアクセスとして知らされます。

また，Human68kではサポートされていませんが，BreakPoint応答サイクルもCPU空間へのアクセスを通して告知されます（図4）。

つまり，基本的にはCPU空間へのアクセスは無視（表1よりFC0 :+: FC1で判別可能）しても構わないが，割り込み応答サイクルだけは(FC2＊FC1＊FC0＊A19)，無視しないということです。いい換えると，どちらかのアクセスだった場合はCPUからX68000へのアクセスです。「どちらか」というのは論理和のことですから，以上をまとめると，

(FC0 :+: FC1)＋(FC2＊FC1＊FC0＊A19)

ということになるわけです。

この信号はX68000へのアクセスというわけですからCONTENと名づけました。

表1 Addres Space Encodings

FC 2	FC 1	FC 0	Adress Space
0	0	0	(Undefined, Reserved)
0	0	1	User Data Space
0	1	0	User Program Space
0	1	1	(Undefined, Reserved)
1	0	0	(Undefined, Reserved)
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

図3-1

図3-2 図3.1のタイミングチャート

図2-1 今回の回路図

図2-2 PLDの等価回路その1（アドレスデコーダ）

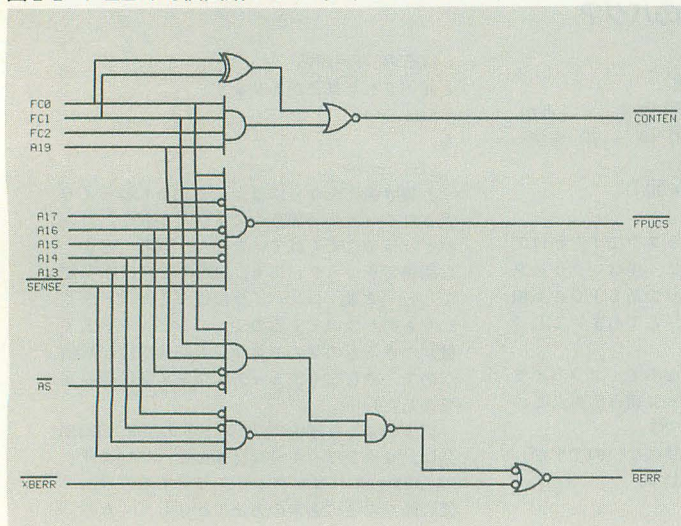
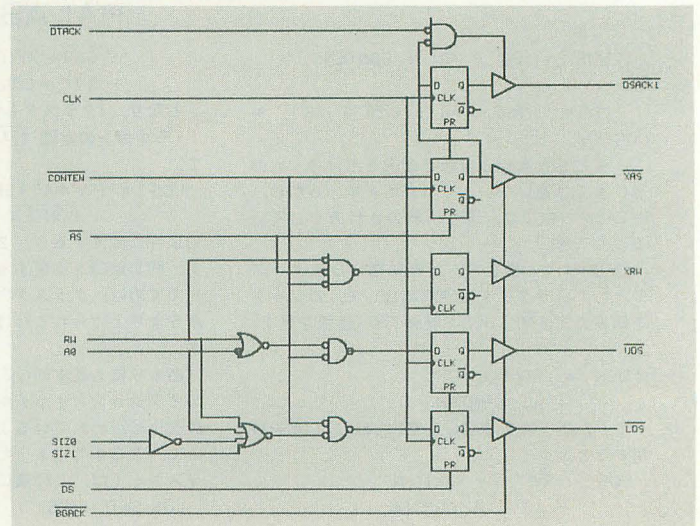



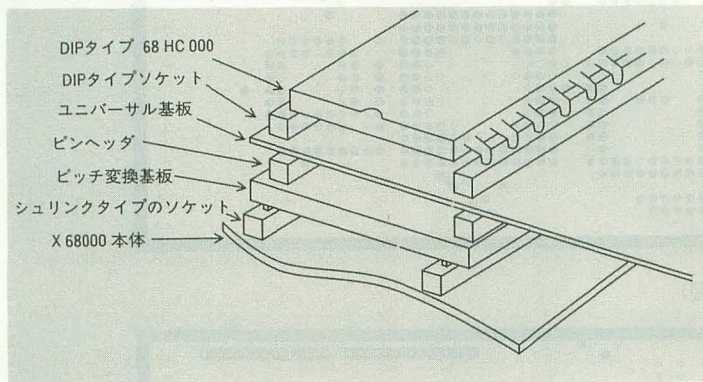
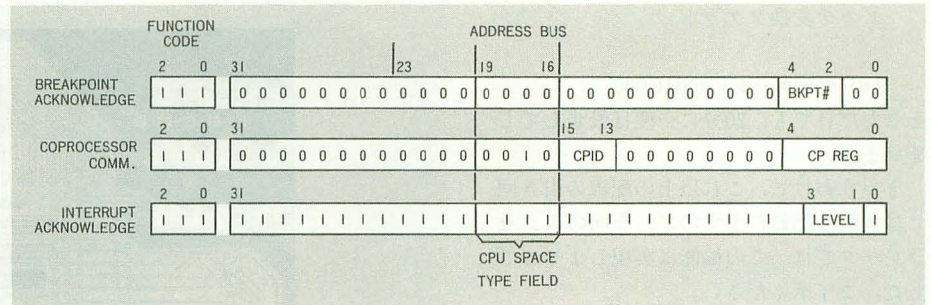
図2-3 PLDの等価回路その2

68000との共存について

パソコンのCPUを取り換えるというのは大変なことです。パソコンのなかで，いちばん大事な部分を取り換えてしまうわけですから，プログラムのなかには不都合の起きるものもあるでしょう。代表例として，ゲームが速すぎて遊べなくなるというものがありますが，そのほかにもキャッシュによる不都合が起きるかもしれません。

そこで，たいていのアクセラレータには，スイッチひとつで従来の環境を復元できるような機能があります。かの「040turbo」もスイッチひとつでCPUを68030に変更できるようです。

最初，わがアクセラレータでもこのような機能を実現しようと思っていました。

図4

```
               FUNCTION
               CODE                          ADDRESS BUS
               2   0  31          23     19   16                      4   2    0
BREAKPOINT    |1 1 1| |0 0 0 0 0 0 0 0 0 0 0 0 0|0 0 0 0|0 0 0 0 0 0 0 0 0 0 0 0|BKPT#|0 0|
ACKNOWLEDGE
               2   0  31                          15  13               4        0
COPROCESSOR   |1 1 1| |0 0 0 0 0 0 0 0 0 0 0 0 0|0 0 1 0|CPID|0 0 0 0 0 0 0 0|CP REG|
COMM.
               2   0  31                                                   3  1 0
INTERRUPT     |1 1 1| |1 1 1 1 1 1 1 1 1 1 1 1 1|1 1 1 1|1 1 1 1 1 1 1 1 1 1 1 1 1|LEVEL|1|
ACKNOWLEDGE
                                                CPU SPACE
                                                TYPE FIELD
```

しかし，写真3のような回路を組んでみて挫折しました。断面から見ると図5のようになっていて，電気的にはなんの変哲もないもので，ただメイン基板からの信号を，

ピッチ変換ソケット

↓

ピンヘッド

↓

プリント基板

↓

ソケット

↓

DIPパッケージの68HC000

とつないだものです。が，すでにこれだけのことで動かなくなってしまうのです。

実験に用いたのは比較的安定していると

図5　CPUを持ち上げただけで動かなくなる……

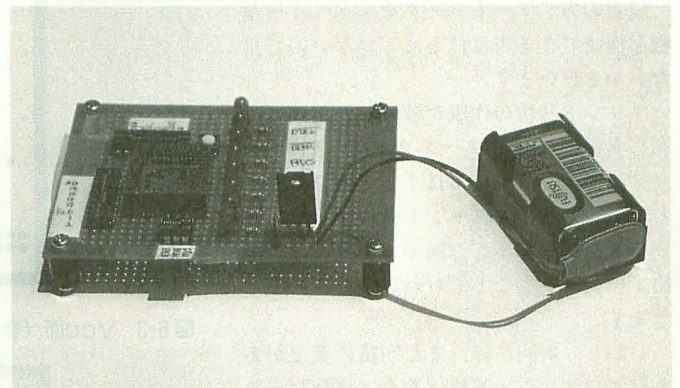



写真3　テスト用回路

リスト1

```
 1: ;PALASM Design Description
 2:
 3: ;---------------------------------- Declaration Segment ------------
 4: TITLE    LUCKY30
 5: PATTERN
 6: REVISION 001
 7: AUTHOR   ISHIGAMI Tatsuya
 8: COMPANY  S.B.C.
 9: DATE     11/18/93
10:
11: CHIP  _LUCKY30  MACH210
12:
13: ;---------------------------------- PIN Declarations ---------------
14: PIN        10      /SENSE                  ; INPUT
15: PIN        11      A13                     ; INPUT
16: PIN        13      A14                     ; INPUT
17: PIN        14      A15                     ; INPUT
18: PIN        15      A16                     ; INPUT
19: PIN        16      A17                     ; INPUT
20: PIN        17      FC2                     ; INPUT
21: PIN        18      FC1                     ; INPUT
22: PIN        19      FC0                     ; INPUT
23: PIN        20      A19                     ; INPUT
24: PIN        21      XRW     REGISTERED      ; OUTPUT
25: PIN        9       /DSACK1 REGISTERED      ; OUTPUT
26: PIN        8       /FPUCS  COMBINATORIAL   ; OUTPUT
27: PIN        7       /BERR                   ; INPUT
28: PIN        6       SIZ1                    ; INPUT
29: PIN        5       /AS                     ; INPUT
30: PIN        4       /DS                     ; INPUT
31: PIN        3       RW                      ; INPUT
32: PIN        2       SIZ0                    ; INPUT
33: PIN        35      CLK1                    ; INPUT
34: PIN        37      /UDS    REGISTERED      ; OUTPUT
35: PIN        38      /LDS    REGISTERED      ; OUTPUT
36: PIN        24      /XBERR  COMBINATORIAL   ; OUTPUT
37: PIN        25      A0                      ; INPUT
38: PIN        26      BGACK                   ; INPUT
39: PIN        27      /CONTEN                 ; NC
40: PIN        28      /DTACK                  ; INPUT
41: PIN        29      /XAS    REGISTERED      ; OUTPUT
42: ;---------------------------------- Boolean Equation Segment ------
43: EQUATIONS
44:
45: ;Active if CPU access Memory or IACK Cycle
46: ;/CONTEN = (FC0 :+: FC1) + (FC1 * FC0 * A19)
47: CONTEN = (FC0 :+: FC1) + (FC2 * FC1 * FC0 * A19)
48:
49: ;Active while FPU(CpID = 1) Access
50: FPUCS  = FC1 * FC0 * /A19 * A17 * /A16 * (/A15 * /A14 * A13) * SENSE
51:
52: ;Active /XBERR or Illegal FPU Access
53: BERR   = XBERR + (AS * FC1 * FC0 * A17 * /A16) * /(/A15 * /A14 * A13 * SENSE)
54:
55: XAS := CONTEN
56: XAS.CLKF = CLK1
57: XAS.SETF = GND
58: XAS.RSTF = /AS
59: XAS.TRST = BGACK
60:
61: DSACK1 := XAS
62: DSACK1.CLKF = CLK1
63: DSACK1.SETF = GND
64: DSACK1.RSTF = /AS
65: DSACK1.TRST = DTACK * XAS
66:
67: ;UDS := (RW + /A0) * CONTEN
68: /UDS := /RW * A0 + /CONTEN
69: UDS.CLKF = CLK1
70: UDS.SETF = /DS
71: UDS.RSTF = GND
72: UDS.TRST = BGACK
73:
74: LDS :=  (RW + A0 + /SIZ0 + SIZ1) * CONTEN
75: LDS.CLKF = CLK1
76: LDS.SETF = /DS                 ;Correctry GND
77: LDS.RSTF = GND                 ;Correctry /DS
78: LDS.TRST = BGACK
79:
80: XRW := /(AS * CONTEN * /RW)
81: XRW.CLKF = CLK1
82: XRW.SETF = GND
83: XRW.RSTF = GND
84: XRW.TRST = BGACK
85:
86: ;---------------------------------- Simulation Segment -----------
87: SIMULATION
88: ;-----------------------------------------------------------------
```

されるX68000EXPERTですが，さらに，

ピッチ変換ソケット

↓

DIPパッケージの68HC000

としただけでも，20分に一度は暴走してしまいます。

というわけで，これ以上の配線の引き回しは危険と思われますので，今回のアクセラレータでは，この機能は実現しませんでした。ご了承ください。

プリント基板について

今回使用したプリント基板のアートワークを図6-1～6-4に示します。写真4には，数カ所にパターンカットなどの事後処理的な部分が見えますが，図6-1～6-4では，すべてデバッグされています。

読者の方でアートワークをもとに4層基板を作成できる方はほとんどいないのではないかと思います。

プリント基板の作成を依頼した工場には，私の名義でアートワークフィルムやNCテープが保管されているはずですので，どうしても，いますぐプリント基板がほしいという方は編集部までご連絡くだされば，価格応相談ということでおわけすることもできます。

しかし，次回には，また回路の変更があるはずです。この連載（そう，連載だったのです）が成功裏に終わった暁には，きちんと配布を行いますので，お急ぎでない方はもうしばらくお待ちください。おそらく，1枚5千円～1万円位で可能なはずです(鋭意努力中)。

ピンヘッドについて

今回作成するアクセラレータは，拡張スロットに差し込むタイプのものではありません。拡張スロットを使用する代わりに，CPUをソケットから引き抜き，そのソケットから直接，信号のやり取りを行います(ですから，ソケットを用いずに基板に直接

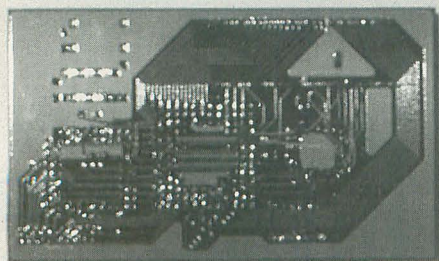



写真4　プリント基板

図6-1　ハンダ面（参考図）

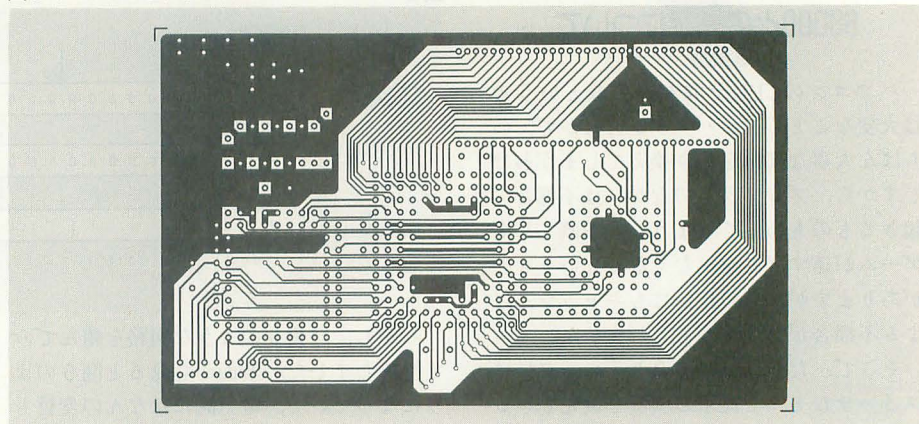



図6-2　GND面（参考図）

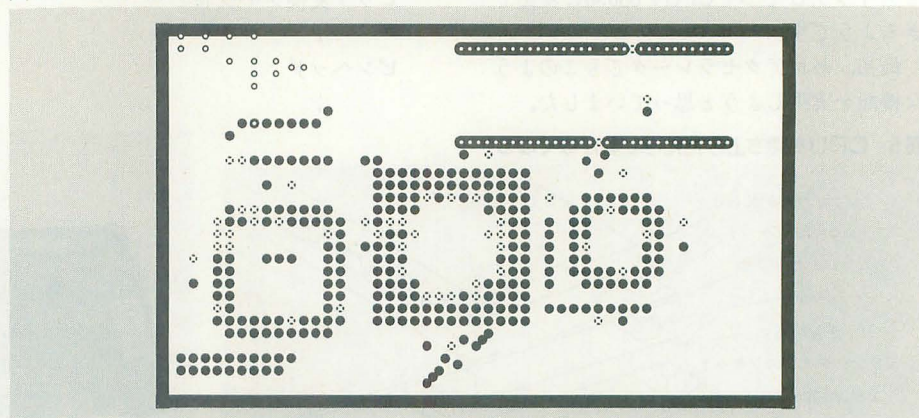



図6-3　Vcc面（参考図）

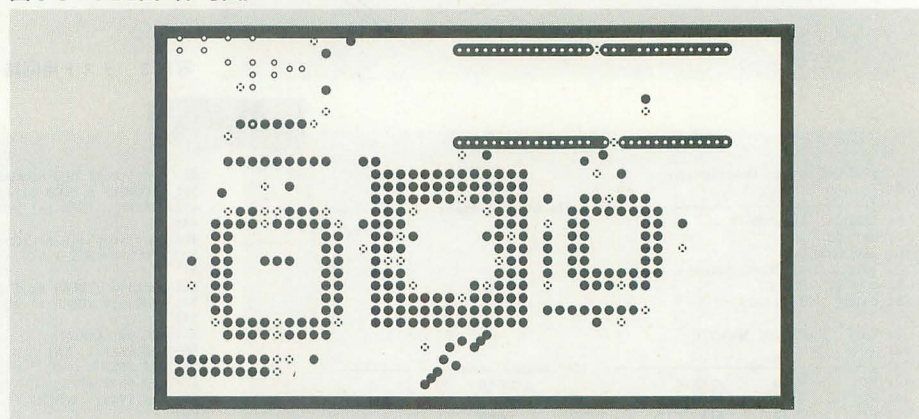
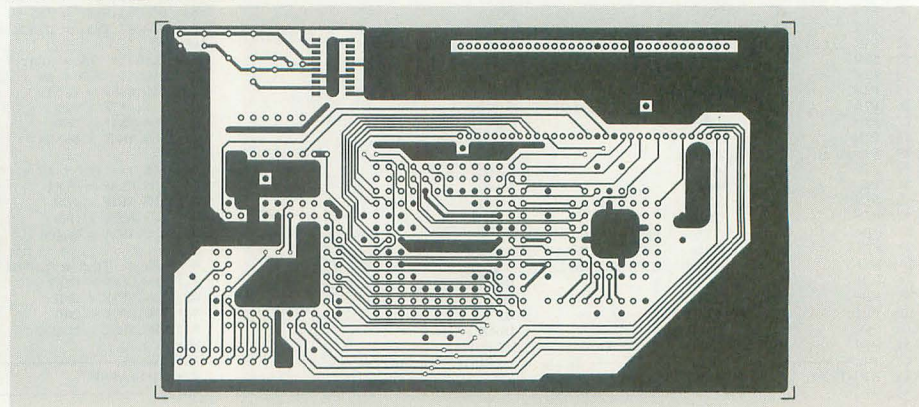



図6-4　部分面（参考図）

CPUをつけているCompactXVIでは使用できません)。

そういうことで，ソケットとアクセラレータの基板とを電気的に接続してやらなくてはなりません。このような場合は，たいてい「ピンヘッダ」と呼ばれる部品を使うのですが，X68000で用いられている68000は「シュリンクタイプ」と呼ばれるピッチ=1.5ミリの特殊なタイプなので，適切なものが見つかりません。

繰り返し述べているように，X68000では，CPU回りの配線を引き延ばした場合，対ノイズ性が微妙なところにあるので，不必要に長いピンヘッダはあまり好ましくありません。

また，ソケットとアクセラレータをただつないでいればいいかというと，そうでもなく，ソケットの回りには，ほかにもさまざまなLSIが並べられていますので，それらの背中に基板が接触してしまうようなことがあってはいけません。ある程度の長さは必要なのです。

ピンヘッダに限らず，ラッピング用の64ピンシュリンクタイプソケットでもよいのですが，秋葉原中探し回っても見つけることができませんでした。私の友人で，どこかで見たことがあるという人がいるのですが，何十年も秋葉原でお店をやっていて見たこともないとおっしゃる方もいるのですから実際のところはわかりません。

そこで，今回は抵抗のいらなくなったリード線を以下のようにハンダづけして，ピンヘッダに代用しました。

いきなり，リード線をハンダづけをしても線先が揃うはずありませんから，それなりの道具を用意します。

シュリンクタイプの64ピンソケットは割合手軽に入手できるようなので，今回はこれを用いました。ピンの横方向へのずれ防止という点から，なるべく丸ピンタイプのものをおすすめします。

まず，4隅のリード線を先にハンダづけします。これを基準としますので，ハンダづけ後は定規などを用いて，各線が同じ長さで基板から垂直についているかをチェックしてください（図7-1)。

この4本のリード線に，ソケットを差し込みます（図7-2)。

基板の上から，ぷすぷすリード線を差し，ハンダづけをします（図7-3)。

64本のリード線のハンダづけが完了したら，垂直にソケットを抜きます。このとき，無理な力をかけたりしてピンを曲げないように注意してください。うまく抜ければ，この部分は完成です。多少曲がっても，もともとは抵抗のリード線ですから，ピンセットで直せます。折れたりしません。

というわけで，ピンヘッドの豊富な品揃えを行っているお店をご存じの方がいらっしゃいましたら，編集部までご連絡ください。

図7-1 まず基板の裏側に4本リード線を立てます

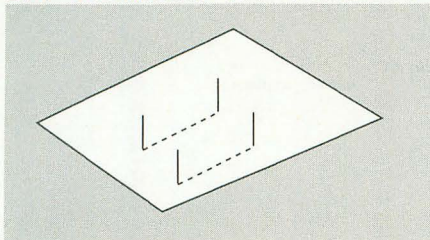



図7-2 シュリンクタイプのソケットをはめこみます

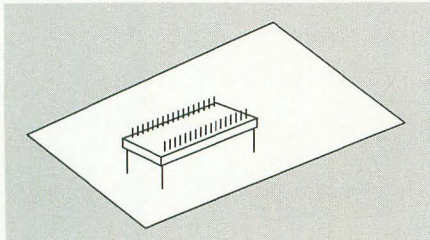



図7-3 ソケットに残りのリード線をはめこみます

ソフトウェアについて

あんなに不安定だったものが意外にもあっさりと動いてしまい，面食らってしまいました。

ひょっとして，これはバスエラーでも起こさない限り68000と68030との相違はHuman68kがすべて吸収してくれるのではないかと，ver.3.0のマスターディスクを突っ込んでみました。すると，fddevice.xを組み込もうとした時点で，

　　エラー($01AC)が発生しました

と出てしまいます。きっと，どこか具合が悪いのでしょう。

じゃあ，キャッシュの威力はいかほどかしらんとcacheコマンドを実行させようとすると，またまた，

　　エラー($01AC)が発生しました

と表示されてしまいます。これはX68000がIOCSコール$ACをサポートしていないために起こるエラーです。

IOCSコール$ACとは，68030のキャッシュ，MMU，FPUの状態を制御するコールで，X68000でサポートされていないのは当たり前です。

というわけで，対処法がリスト2です。このプログラムを実行するとIOCSコール$ACがX68030とほぼ同等にサポートされます。別にたいした処理内容ではありませんので，図8を見ながら流れを追いかければ，なにをやっているのかわかると思います。

図8 CACRレジスタ

31–14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00000C000000000000	WA	DBE	CD	CED	FD	ED	0	0	0	IBE	CI	CEI	FI	EI

WA=Write Allocate
DBC=Data Burst Enable
CD=Clear Data Cache
CED=Clear Entry in Data Cache
FD=Freeze Data Cache
ED=Enable Data Cache
IBE=Instruction Burst Enable
CI=Clear Instruction Cache
CIE=Clear Entry in Instruction Cache
FI=Freeze Instruction Cache
EI=Enable Instruction Cache

リスト2

```
 1: *
 2: *  アクセラレータ用IOCSパッチ
 3: *
 4: *
 5:     .include        doscall.mac
 6:     .include        iocscall.mac
 7:     .text
 8:
 9: FPUFlag    equu     $000cbd
10:
11: CACHE      equ      $ed0073
12:
13: stapr:     bra      endpr
14:
```

数値演算コプロセッサ

いきなり回路が動くようになったので，調子にのって数値演算コプロセッサを接続しました。

X68000では数値演算機能を増加する機能を「数値演算プロセッサ」が担っていましたが，アクセラレータではせっかく68030を使うわけですから「数値演算コプロセッサ」を使うことにします。

前者と後者の違いについては，図9-1を見てください。

数値演算プロセッサが，CPUの外部にあるのに対し，数値演算コプロセッサは，CPUの内部に潜り込み，CPUそのものの機能を拡張します。潜り込むといっても，物理的にパッケージの中に潜り込むわけではありませんが，プログラマから見ればいままで使えなかった命令が使えるようになったりするわけです。

数値演算コプロセッサと68030との接続は図9-2のように行います。

実際に図9-2のように接続してみたところ，最初はまったく動かなかったのですが，データバスに10kΩのプルアップ抵抗，$\overline{\text{DSACKx}}$に4.7kΩのプルアップ抵抗をそれぞれ挿入したところ，ちゃんと動作するようになりました(図2-1の回路図では，ちゃんと修正されています)。

数値演算コプロセッサの認識

数値演算コプロセッサはコンピュータの動作に絶対に必要というものではありません。あれば（小数部を含む）数値演算が高速化されますし，なければ高速化されないといったものです。高速化されないといっても，小数の計算は位取りや符号に注意してやれば，自然数の問題になってしまうわけですから，時間をかければCPU単体でもなんとか計算できるわけです。

数値演算を行う際，Human68k上ではデバイスドライバFLOAT2を組み込んでおけばCPU単体で，デバイスドライバFLOAT4を組み込んでおけば数値演算コプロセッサとCPUが共同で計算を行うようになっています。

これと同じように，アクセラレータのハードウェアのほうでも数値演算コプロセッサが組み込まれていた場合とそうでない場合について，2とおりの動作を用意しなければなりません。

数値演算コプロセッサがあるのに，これ

```
15: iocsAC:
16: ****************************************************************
17: *IOCS $AC system status
18: *  D1 : MODE
19: *           0       MPU status
20: *           1       cache status
21: *           2       cache defalut
22: *           3       cache flush
23: *           4       cache cntrol
24: *
25: *  D2 : cache bit
26: *           bit 0   instruction cache
27: *           bit 1   data cache
28: *
29: ****
30: *
31: sys_stat:: movem.l d1/d2,-(sp)
32:            moveq   #-1,d0                   for error code
33:            cmp.w   #4,d1
34:            bhi.s   mode_err
35:            add.w   d1,d1
36:            moveq   #0,d0
37:            move.w  jobtbl(pc,d1.w),d1
38:            jsr     jobtbl(pc,d1.w)
39: mode_err
40:            movem.l (sp)+,d1/d2
41:            rts
42:
43: jobtbl
44:            dc.w    mpu_stat-jobtbl          0
45:            dc.w    cache_stat-jobtbl        1
46:            dc.w    cache_defalut-jobtbl     2
47:            dc.w    cache_cicd-jobtbl        3
48:            dc.w    cache_ctrl-jobtbl        4
49:
50: cache_stat
51:            dc.w    $4e7a,$1002                  *movec  CACR,d1
52:            ror.l   #1,d1                    EI(31) ED(7)
53:            lsr.w   #7,d1                    EI(31) ED(0)
54:            rol.l   #1,d1                    EI(0)  ED(1)
55:            and.w   #3,d1
56:            move.w  d1,d0
57:            rts
58:
59: cache_defalut
60:            move.b  CACHE,d2
61:            bsr     cache_ctrl
62:            rts
63:
64: cache_cicd
65:            dc.w    $4e7a,$0002              *movec  CACR,d0
66:            or.w    #$0808,d0
67:            dc.w    $4e7b,$0002              *movec  d0,CACR
68:            and.w   #$f7f7,d0
69:            dc.w    $4e7b,$0002              *movec  d0,CACR
70:            rts
71:
72: cache_ctrl bsr     cache_stat
73:            and.w   #3,d2
74:            moveq   #0,d1
75:            add.w   d2,d2
76:            move.w  cache_reg(pc,d2.w),d1
77:            dc.w    $4e7b,$1002              *movec  d1,CACR
78:            rts
79:
80: cache_reg  dc.w    $0000
81:            dc.w    $0001
82:            dc.w    $2100
83:            dc.w    $2101
84:
85: * MODE 0
86: *  D0 : high word
87: *          clock speed (MHz x 10)
88: *
89: *       bit 15 FPU
90: *          0 : なし
91: *          1 : あり
92: *
93: *       byte MPU
94: *          3 : 68030
95: *
96: * MODE <> 0
97: *  D0 : cache mode
98: *
99: mpu_stat
100:           move.l  #$00030003,d0
101:           rts
102:
103:
104:    .even
105: endpr:
106: * FPUありの場合
107:    move.l  #FPUFlag,a1
108:    move.b  #1,d1
109:    IOCS    _B_BPOKE
110:
111: * iocsACを登録する
112:    pea     iocsAC
113:    move.w  #$1ac,-(sp)
114:    DOS             _INTVCS
115:    addq.l  #6,sp
116:    clr.w   -(sp)
117:    move.l  #endpr-stapr,-(sp)
118:    DOS             _KEEPPR
119:    rts
120:
121:    .end
```

を無視しCPU単体だけで頑張ろうとする場合は特に問題ありません。逆に数値演算コプロセッサがないのに，CPUが共同作業を行おうとしたときに問題が発生します。CPUが数値演算コプロセッサに対し共同作業を申し出ます。しかし，数値演算コプロセッサはいないわけですから，返事はいつまでたってもくるわけありません。それでもCPUはいつまでも返事を待ち続けます。CPUはパソコンの頭脳ですから，この場合，X68000はまったく動かなくなってしまいます。

こんなときには外部でバスエラーを発生させてやります。

68882には$\overline{SENSE}$という端子があり，常に0Vの電圧を出力(?)しています。つまり，この信号をプルアップしておけば，

数値演算コプロセッサあり
→SENSE＝0 （$\overline{SENSE}$＝1）
数値演算コプロセッサなし
→SENSE＝1 （$\overline{SENSE}$＝0）

となるわけです。

CPUとX68000本体の切り放し

また，「アクセラレータを作るからには，数値演算コプロセッサを接続すべき。コプロセッサとの通信中はX68000とCPUとの接続を遮断しなければならないので，データバスにバッファを挟むべきではないか」というようなご意見を読者の方からいただきました。

一般的にいって，68000のソケットに差すタイプのアクセラレータではこのような処置が必要です。

図10-1はX68000のCPU付近の回路図ですが，CPUを出たデータバスは各所に接続される前にいったん74AS245を通っています。74AS245とは，バッファ機能を実現するICなのですが（図10-2に内部等価回路図），これを制御できれば，CPUとX68000との接続を切ることができます。

74AS245は$\overline{G}$＝1になったとき両ポートを電気的に接続するのですが，常識的にいって両ポートを接続して意味のあるときのみ行われると考えてよいでしょう。つまり，ポート上に有効なデータが乗っている場合です。74AS245のポート上には，データバスが乗っています。このデータバスが意味のあるデータを示している場合には，68000の$\overline{DS}$(Data Storobe)端子が1になってます。

つまり，CPUと数値演算コプロセッサの通信中はこの$\overline{DS}$端子を0にしてやればよいのです。

実際，リスト1では$\overline{AS}$，$\overline{UDS}$，$\overline{UDS}$，ともに$\overline{CONTEN}$＝1でないと有効にはならないようになっています（それぞれ，55,68,74行）。

アドレスデコーダ

図2-1を見ると構成部品が3つあります。CPUと数値演算コプロセッサはよいとして，アドレスデコーダと書かれた部品があります。

68030では，8つまでのコプロセッサを自分の分身としてメモリ空間とは別なアドレス空間に配置し，アクセスすることが可能となりました。8つのコプロセッサは，それぞれ0～7のIDで管理され，数値演算用のコプロセッサ（688881/2）は，ID＝1とモトローラによって決められています（ちなみに，ID＝0はメモリ管理用のコプロセッサで，その他は未定）。

CPU空間アクセス時における68030のアドレス情報は表1のようになっていますから，これを順にデコードしていけば図2-2にあるようなアドレスデコーダが完成するわけです。

まず，CPU空間アクセスである場合を抜き出します。

FPUCS＝FC2＊FC1＊FC0

が表1.1をよく見ると，

FC2＝0
FC1＝1
FC0＝1

というのは使われないことになっています

図 9-1 数値演算プロセッサと数値コプロセッサの違い

(a)数値演算プロセッサの場合
通常のメモリ空間に置かれ，他のデバイスと同様の扱いを受けます

(b)数値演算コプロセッサの場合
FPUはCPUの一部となりCPUの機能を拡張します

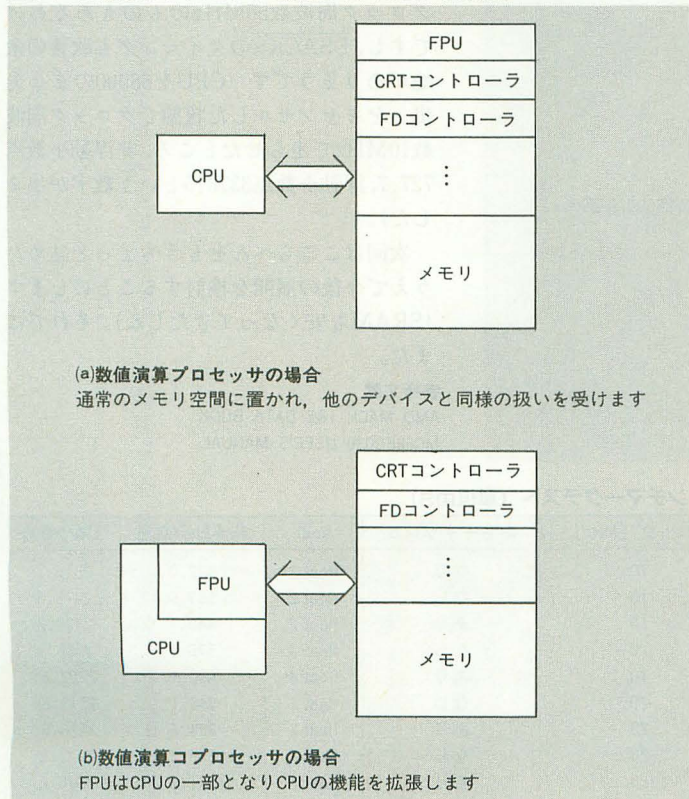
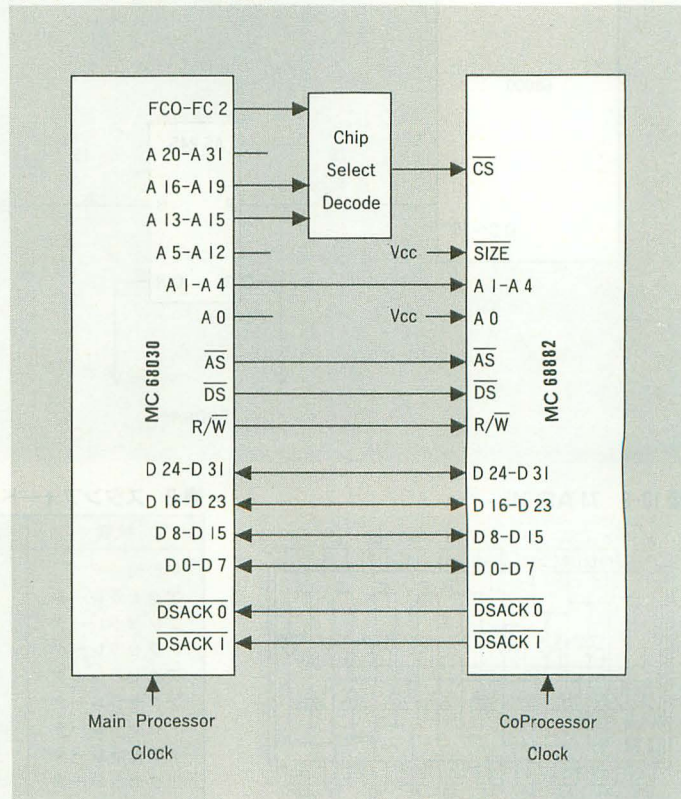



図 9-2 コプロセッサの接続

から，

$$FPUCS=FC1*FC0$$

でCPU空間アクセスであることがわかります。

コプロセッサとの通信は真ん中のCO PROCESSOR COMMモードで行われますから，

$$FPUCS=FC1*FC0*\overline{A19}*\overline{A18}*A17*\overline{A16}$$

ですが，これも図9をよく見ると，常にA19＝A18ですので，

$$FPUCS=FC1*FC0*\overline{A19}*\overline{A16}$$

となります。

前述のコプロセッサID＝1という約束を思い出して(A15＝0，A14＝0，A13＝1)，

$$FPUCS=FC1*FC0*\overline{A19}*\overline{A16}*\overline{A15}*\overline{A14}*A13$$

となります。

図2-2では，まだ「CP REG」の部分が解決されていませんが，これはコプロセッサのどのレジスタをどうするかを示す信号ですので，アドレスデコーダ自体には無関係です。*A4～0は直接コプロセッサに接続します。*

つまり，ちゃんと数値演算コプロセッサを使えるのは，アドレスが一致して実際にソケットに68882が差されている状態ということですから，数値演算コプロセッサに与える$\overline{FPUCS}$(FPU Chip Select)信号は，

$$FPUCS=FC1*FC0*\overline{A19}*\overline{A16}*\overline{A15}*\overline{A14}*A13*SENSE$$

となります。

逆に，コプロセッサを持たないくせにアクセスしようとしたというのはアドレス情報が有効であることを示す$\overline{AS}$とあわせて，

$$FPUCS=AS*FC1*FC0*\overline{A19}*\overline{A16}*\overline{(\overline{A15}*A14*A13*SENSE)}$$

となります(ID＝1番以外は許しておきましょう)。

本来のバスエラーを示す$\overline{XBERR}$とあわせて，アクセラレータ上でのバスエラーを示す$\overline{BERR}$は，

$$BERR=XBERR+AS*FC1*FC0*\overline{A19}*\overline{A16}*\overline{(\overline{A15}*A14*A13*SENSE)}$$

となります。

このような回路を組み込んでおけば，数値演算コプロセッサが装着されているか否かにかかわらず正しく起動できます。

図10-1　X68000のデータバス

図10-2　74AS245

どのくらい速くなるのか

アクセラレータということで気になるのは，やはり実行速度の向上です。部品代と馬鹿になりませんし，せっかく苦労して取り付けても，さほど性能が上がらないのでは話になりません。

Oh!Xでよく使用されているスタンフォードベンチマークプログラムの結果が表2です(中森章氏作成のstanf_v21.xを使用)。

現時点ではデータキャッシュをonにするとなぜかベンチマークプログラムが暴走してしまいますので，キャッシュは命令キャッシュのみをon/offしています。

表中のクロック20MHzというのは，今回の回路に若干の変更を加えて本体とCPUが完全に非同期動作するように改造したときのものです。20MHzとはいっても，まだ，あまり練っていないのでX68000側の$\overline{DTACK}$先出しを完全にキャンセルしていますし，さらに，68030側でもデータの受け取りに1クロック分，余計なサイクルが入ってしまいます。

表の最上列と最下列とを比較すると約1.6倍になっているわけで，とりあえずEXPERTをXVI相当にすることには成功したといえそうです。データキャッシュの不具合が解決すれば2倍速はいくでしょう。

今回は手が回りませんでしたが，68030はクロック周波数50MHzのものもあるわけですし，$\overline{DSACKx}$のタイミングも改善の余地がありそうです(CPUを68000のまま先出しをキャンセルした状態でクロック周波数10MHzで走らせたところ，非浮動小数点727.7，浮動小数点3370.5という数字が出ました)。

次回はここらへんをもうちょっと詰めたうえで今後の展開を検討することにします(SRAMも安くなってきたしね)。それではまた。

参考文献
AMD MACH 1&2 DATA BOOK
MC68EC030 USER'S MANUAL

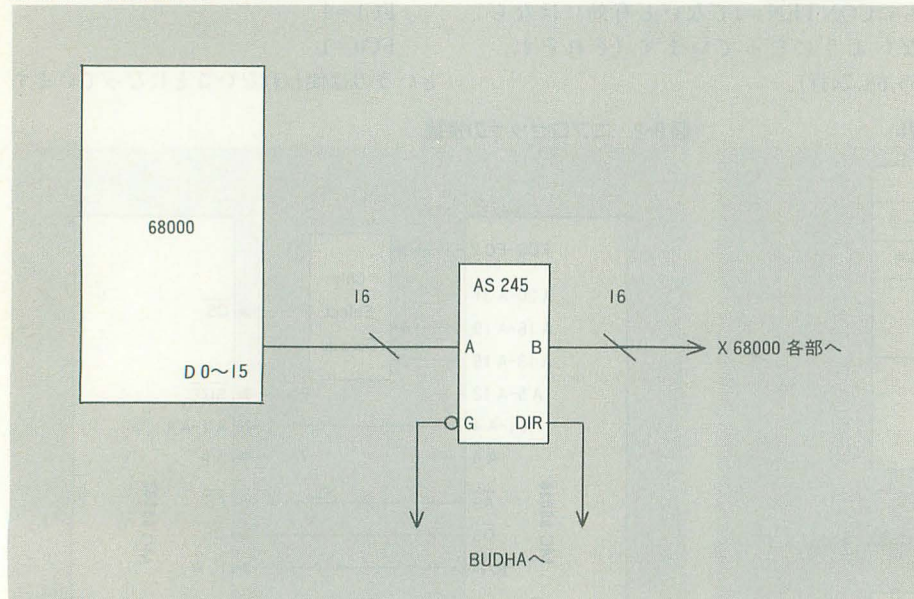



表2　スタンフォードベンチマークテスト(単位ms)

状態	クロック［MHz］	命令キャッシュ	float	非浮動小数点	浮動小数点
無改造	10	なし	float 2	572.0	2516.7
アクセラレータ	10	なし	float 2	597.0	2776.3
アクセラレータ	10	あり	float 2	440.1	2430.7
アクセラレータ	10	なし	float 4	576.1	2688.3
アクセラレータ	10	あり	float 4	430.4	2287.8
アクセラレータ	20	なし	float 2	548.1	2519.8
アクセラレータ	20	あり	float 2	368.3	2138.6
アクセラレータ	20	なし	float 4	529.2	2323.5
アクセラレータ	20	あり	float 4	361.7	1873.2

ローテク工作実験室　第4回

内蔵AD PCM高音質化計画

Taki Yasushi 瀧　康史

今月のローテク講座はいきなり本体改造に入ります。X68000のAD PCMをクロックアップします。一部，ハイテクですのでちょっと注意が必要です。あくまでも個人の責任において実行してください。

あちらでもこちらでも，そろそろX68000の内蔵音源については，とやかくいわれるようになりました。7年前は最新のスペックだった機能も，いまとなってはすでに，過去の遺物？　ただ，X68000を愛しすぎて，妙なスペックを望んでいる人がいることも事実です。

次のマシンではPCM16音＋FM16音ほしいとか，なかにはむちゃくちゃな意見を出している人がいます。よぉく考えてみると，FM音源チップはデータ量が少ないという利点のみで，標準メモリ12Mバイト*1，移動メディアはMOが平均的な*2現在では，あまり意味はありません。

なにより，FM音源チップは高いのでPCMチップを適当に使うのが妥当です。PCMも現在では，8ビット24kHzぐらいあればなんとかなりそうですが，きっと1年後ぐらいには，48kHz16ビットができないと「カス」とか「使えない！」とかいわれてしまうでしょう。

そういうことでPCMボードを作っているのですが，これがコントローラからなにから起こしているので，かなり時間がかかりそうです*3。こちらのほうはおいおいやっていくことにしましょう。でも，やっぱりそれまでの「つなぎ」がほしいところです。簡単なPCMボードを作ってもよいのですが，それでは自分が納得できません。それに，あとからよいボードを出してもユーザーの懐に問題がありそうですし。

そこで，安上がりに，X680x0本体を多少改造する程度で，PCMが高音質にならないか考えてみました。

＊1 別に標準実装メモリといっているわけではなく，標準推奨メモリといっている。
＊2 無作為に私の友人を探ってみたところ，10人に8人が120MバイトMOユーザで，1人が230MバイトMOユーザーでした。
＊3 PCMボードに関して興味がある方は，私が主催しているPCVAN内のXICLUBにきていただければ幸いです。ちなみにSIGは無料です。

MSM6258Vとは

X680x0が内蔵しているPCM音源は沖電気のMSM6258というICです。ある筋の話によると，これは留守番電話などに使われるもののようで性能もその程度にすぎません。

しかし，1チップで録音/再生ができ，しかも分解能10ビットのデータをたったの4ビットにして，それでもソコソコの音質をキープするわけですから，なかなか高性能なICだといえます。メインメモリ1Mバイトの初代機が出た当時の背景からこのことを考えると，なかなか賢明な選択だったと思われます。

ところでサンプリングレートを一定にするためには，このICにはある基調となるクロックが必要になります。その基調周波数はX680x0では8MHz/4MHzと選択できます（YM2151のレジスタ$1Bで選択）。15.6kHz，10.4kHz，7.8kHzという中途半端な数字はこの基調周波数から算出していて，これはそれぞれ基調周波数を1024分周，768分周，512分周にしているだけです。

MSM6258にはこの3つの分周モードしかなく，このままではX68000には3モードしかできあがりません。そこで4M/8MHzの2つのクロックを選択するのです。単純に考えると6モードできそうな気がしますが，8MHzのときの1024分周と4MHzのときの512分周は同じなので，結果として，X680x0は5モードのPCMレートを持つことになります。

改造の本筋

X680x0では8MHzの周波数をMSM6258に加えます。仮に8MHz以上の周波数を加えたら，当然ながら音は高くなるはずです。今回のローテクの主眼は，この基調周波数を変えるところにあります。

とりあえず12MHzのオシレータを入れてみたところ，予想どおり音が高くなりました。16MHzではさらに高くなります。

MSM6258とそれに関する回路は，マンハッタンシェイプのX680x0の場合，すべて下面基板に搭載されています。基調周波数の4M/8MHzは，下面基板に搭載されている16MHzのオシレータから2分周，4分周しているようです。

基板を剥き出しにしてみたところ，この16MHzのオシレータはわりと近くにあります。ですから，この16MHzをMSM6258に入れれば，ほかにオシレータを購入する必要はありません。

16MHzを入れた場合，計算上，512分周で31.25kHz，768分周で20.83kHz，1024分周で15.625kHzと割と妥当な組み合わせになります。当然，入力も出力もレートが上がりますから，この改造を行えば31.25kHzで録音し再生することもできるようになるわけです。

しかしながら，このままでは互換性がなくなってしまい，従来のデータからゲームからすべて周波数が2倍に上がってしまいます。そのために，コントローラICをつけ，ゲームや，データを聞く分には問題ない程度に，従来との互換性を保つことにします。これらの詳細は，この先のコントロ

ーラの章で詳しく書くとして，ここではとりあえず，改造の本質はこれで終わりだといっておきましょう。

入力フィルタ

PCMのICの出力レートを変えたところで，そのままでは具体的な音質向上にはなりません。

その理由はフィルタ回路にあります。

入出力段階でのフィルタ回路は音に対して余計な部分，つまりノイズを除去したりするためのものです。また，オーディオケーブルを通って変な直流電流が加わり，回路に負荷をかけないようにするという意味などもあります。

どちらにしてもよい音で聞くためにも音声回路にフィルタ回路は必需品で，ないと故障の原因にもなるものです。

さて，MSM6258の入力部分にどのようなフィルタ回路がついているか見てみました。まずは入力フィルタ，すなわち録音時に利用するフィルタ回路です。機種により微妙に違いますが，Outside X68000, X68030 Inside/Outで見たところ，だいたい図1のようになっています。初期型からX68030まで大きく変わった様子はないので，特性はほぼ同じだと考えて構いません。

図1　入力フィルタ

図2　標準状態の周波数特性

図3　図2の拡大図

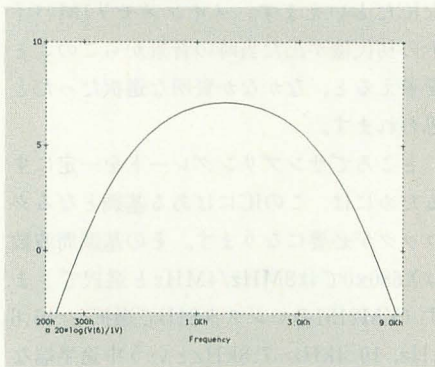



図4　フィルタ調整後の周波数特性

図5　図4の拡大図

この回路で，オペアンプを理想的なものと仮定し，P-SPICEを用いて周波数特性を計算してみました。それが図2です。

AUX入力の正確な値は調べるのが面倒だったので，1Vp-pと仮定します。図2では横軸が周波数，縦軸はdBです。これは全体図で細かいことが把握しづらいので，-3dBに落ちる部分，つまりカットオフ周波数を見てみることにします。それが，図3です。

見たところ，だいたい，200Hz～9kHzまでがバンドパスされています。200Hzでは，ベース，バスドラムの音はうまく通りません。しかし，MSM6258本来の目的である，「人間の音声」の部分である周波数はうまくブーストされています。最大周波数も9kHz弱，サンプリング定理から，15.6kHzのPCMでは7.8kHz（サンプリング周波数の半分）が最大周波数なので，だいたいよいと思われます。

31.25kHzにする場合，最大周波数が15.6kHzになります。したがって，カットオフ周波数を，上は15.6kHzまで伸ばさねばなりません。サンプリングビットが10ビット程度なので，低音はうまく鳴らないでしょうが，それでも，まったく切ってしまうのはいやだったので，とりあえず50Hzぐらいまでサポートすることにします。

そこで，フィルタ回路に調整を加えます。実際に取り替える部分は図1の回路の，C2, C3, C4です。ここでは，C2, C4を4.7μFに，C3を820pFに取り替えてみました。

取り替えた場合の周波数特性が図4です。図4は図2とレンジが同じなので比べてみて構いません。細かい部分がわからないと思うので，これは図5に表します。

この図によりわかると思いますが，だいたい40Hz～20kHzまで入力できるようにフィルタの特性を変えてみました。

余談ですが，40Hzはすでに人間に聞こえるレベルではありません。もちろんヘッドホンでは聞こえませんし，径が小さなスピーカーでは鳴りません。近くにいると肌をビリビリさせることができるような大きなスピーカー（ウーハー）じゃないと，感じることができないレベルです。

また，20kHzは成人男子で聞こえる人は

ごくわずかというところです。今回の改造では15.6kHzが最大ですから，だいたいノーマルカセットテープの最大周波数程度だといえるでしょう。

出力フィルタ

出力フィルタは再生時に使用するフィルタです。図6がその回路ですが，この回路の出力あとにFM音源との合成回路があるので，実際はもう少し違った特性のフィルタになると思われます。

この回路の周波数特性は図7です。だいたいを把握して，拡大図の図8を見てください。これを見るとカットオフ周波数は約3kHzだということがわかります。つまり，サンプリング定理から割り出される無改造15.6kHzモードでの最大周波数は7.8kHzですから，このフィルタは15.6kHzモードには向いていないフィルタだと思います。

PCM音源がこもっている理由はこういったところにあります。多分，初代X68000が出た当初は，音楽用にこのPCMが使われるとは思っていなかったでしょうし，メインメモリだって1Mバイトでしたから，PCMは7.8kHzが主流に使われると思っていたのでしょう。その結果，7.8kHzモードにあわせたフィルタ特性になったのではないでしょうか。のちのロットでこのフィルタ訂正をしなかったのは，多分，互換性維持のためだと思われます。

今回の改造では31.2kHzPCMなのですから，カットオフは15.6kHz付近でなくてはなりません。そこで図6中の抵抗，R1，R2，R3をそれぞれ13kΩにしてみました。

この改造を施した結果が図9です。拡大図は図10，だいたい15kHzあたりがカットオフ周波数になっていますから，今回の改造では妥当な改造になったといえます。

この改造を行うと通常のPCMも高域が伸びるようになります。その代わり，7.8kHzモードでは量子化ノイズがバリバリ乗ってしまいます。PCM8を常駐させておけば量子化ノイズはなんとかごまかせますし，主に使われる15.6kHzの音は高音が通ることになり，特性が明らかに耳で聞いてわかります。ハイハットなども多少ノイジーですが，抜けるように高い音が聞こえます。

ただ，PCM8を入れることができないゲームなどで，7.8kHz以下のPCM再生されると，まともな音が鳴らなくなってしまいます。それはもちろん，基調周波数4MHzモードがないせいです。PCM8を入れるとどのようなレートのPCMでも15.6kHzで再生するので，この改造を行っても，3.9kHz，5.2kHzのPCMを出力することができます。

最後のノイズ

この章はAD PCMの動作原理を知らない方は見ないほうが賢明でしょう。

サンプリング定理から，最大周波数は，サンプリング周波数/2と求められます。しかし，MSM6258の場合，この最大周波数に必ず方形波上のノイズが乗ります。再生すると高周波にピーというノイズが聞こえているはずです。

これは，Adaptive Differencial変換ノイズでして，MSM6258がAD PCMからPCMに変換をしたときに出るノイズです。具体的にこのノイズの名称がわからないので，Adaptive Differencial Transform Noiseの頭文字から，以後，ADTNということにします。

ADTNがどうしても鳴ってしまう理由は，MSM6258の圧縮アルゴリズムにあり

図6 出力フィルタ

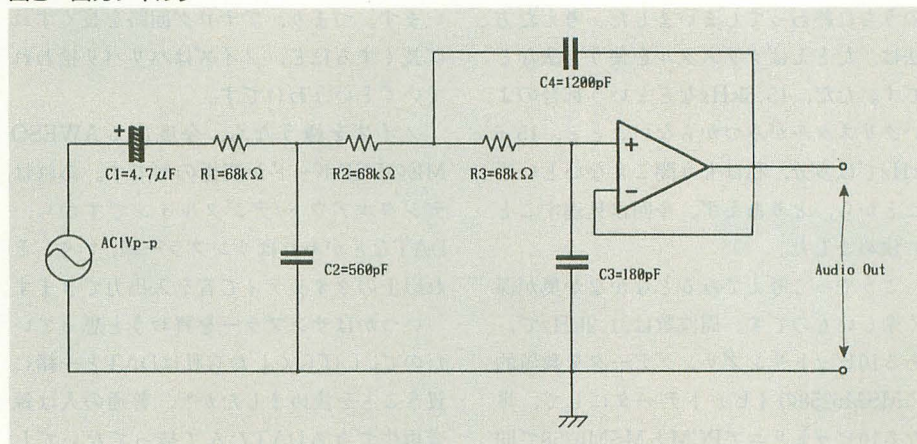



図7 標準状態の出力特性

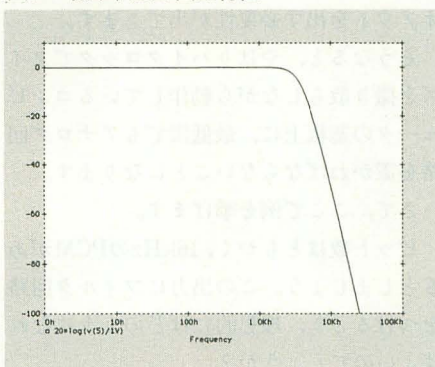



図8 図7の拡大図

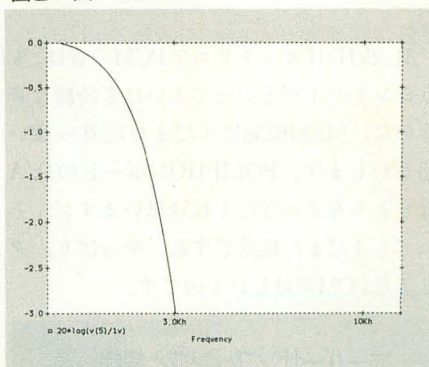



図9 出力フィルタの変更例

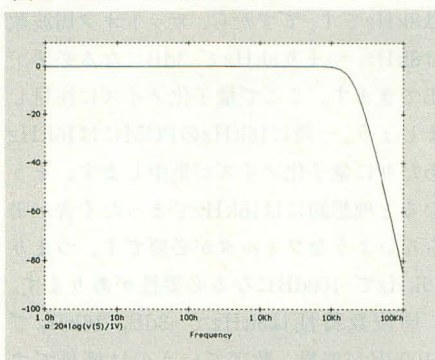
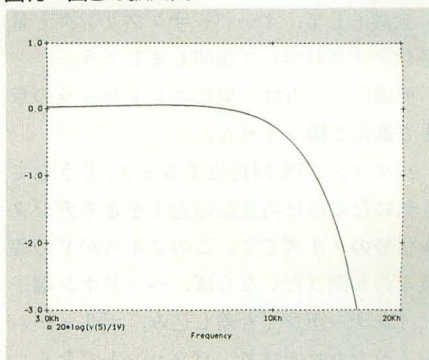



図10 図9の拡大図

ます。たった4ビットのデータ量で，大きな振幅の音を鳴らすために，微小な波形の変移を見逃すからです。ヤマハの音源チップ，OPNAの内蔵PCMは，割合MSM6258のPCMに似ていますが，このOPNAは大きな振幅を見逃す分，振幅0という状態があります。そのあたりが，楽器としてのOPNA，音声サンプリングのためのMSM6258といったところです（ただ，シンバル系の音などでは，私はOPNAよりMSM6258のほうが好きです）。

振幅0がないということは，無音時には必ず最大周波数でADTNが鳴ります。

今回，このノイズをなんとかして取ろうとあがきましたが，時間がないので，構想のうちに終わってしまいました。考えた方法は，たとえばクリスタルを使う方法などです。ただ，15.6kHzなどという都合のよいクリスタルがみつからないことと，15.6kHzでは多分，私はもう聞こえないということから，とりあえず，今回は見逃すことに決めました。

こうやって考えてみるとなかなか奥が深く楽しいものです。周波数は31.2kHzで，ある10ビットサンプリングデータを数値的にMSM6258の4ビットデータにして，単なる10ビットリニアPCMとMSM6258で同じ出力フィルタを使って聞き比べてみました。

31.2kHz10ビットリニアPCMの音は，もうシンセサイザといってよいほど綺麗な音なのに，MSM6258はまだまだ玩具っぽい感じがします。POLIPHONボードのD/Aなどよりもよい音だと私は思いますが，それでもまだまだ玩具ですね。やっぱり，クリアなPCMがほしいものです。

オーバーサンプリングと補間処理

余談として，オーバーサンプリングと量子化ノイズについて説明しましょう。

興味のない方は一気にコントローラの章まで進んで構いません。

パソコンでPCM再生するとき，どうしても気になるのは再生部の近くを走るデジタル信号のノイズです。このノイズがどの程度なのか聞きたいならば，ヘッドホン端子にインナーホンでも差し込み，ボリュームをそれなりに上げて，パソコン上でなんらかの作業をしてみましょう。なにかに同期して，ノイズが強くなったり，弱くなったりします（たとえば，ソフトウェアキーボードを出したり消したりする）。

これはパソコン自体がノイズを出しているからです。これらを除去するためにフィルタがあります。ただし，厳密に「乱数に近い」ノイズ混じりの音から，必要な音だけを抜くという技術はまだ完成していません。結局，ローパス（低音域だけを通すフィルタ）などを利用する程度で，可聴域にもノイズはやはり乗ってしまいます。

これらのノイズはデジタル回路の段階で乗っているのではなく，D/Aコンバータを通してアナログになった「あと」に乗っています。つまり，アナログ回路を長くすれば長くするほど，ノイズはバリバリ拾われていくというわけです。

ノイズを嫌うなら，今度出るAWESOMEのDSPボードが究極の形です。あれはデジタルアウト/デジタルインですから，DATなどがあればサンプラー級，いや，それ以上のクオリティで音を入出力できます。

いつかはサンプラーを買おうと思っていたので，しばらくしたら私はDATと一緒に買うことを決めましたが[*4]，普通の人は録音再生できるDATなんて持ってないでしょうから，やっぱりパソコンからオーディオアウトを出す必要性が出てきます。

そうなると，やはりハイクロックでノイズを撒き散らしながら動作しているコンピュータの基板上に，最低限でもアナログ回路を置かねばならないことになります。

さて，ここで例を挙げます。

ビット数はともかく，16kHzのPCMがあるとしましょう。この出力にフィルタ回路をつけるとき，理想的にはどのようになればよいのでしょうか？

まず，サンプリング定理から最大周波数は8kHzです。ですから，カットオフ周波数は8kHz，つまり8kHzで-3dBになる必要が出てきます。ここで量子化ノイズに注目しましょう。一般に16kHzのPCMには16kHzあたりに量子化ノイズが集中します。そうなると理想的には16kHzでまったく音が鳴らないようなフィルタが必要です。つまり16kHzで-100dBになる必要性があります。

周波数特性は8kHzで-3dB，16kHzで-100dBが理想。数字でいうのは簡単ですが，これを実際にアナログフィルタ回路で行うと，実に16段ぐらいのフィルタが必要になってしまいます[*5]。

16段のフィルタまでいくと，場所も取る分，ノイズも乗りやすくなります。しかもアナログ部品は割合値段に比例し，よい音で鳴らすためには高くなるので，コストもバカになりません。

通常，パーソナルコンピュータについている音源はこのあたりをいい加減に処理して，安価にしているようです。アナログ部分は3段ぐらいに抑え，上のような場合では技術者のデザインにもよりますが，だいたいにおいて6kHzあたりで落ちはじめ，16kHzぐらいで-30〜-50dBになるフィルタを利用します。

しかし，オーディオ専用機となるとそうもいきません。きっちり8kHzで落ちるべきで，量子化ノイズもきっちり取らねばなりません。そこで，デジタルフィルタを利用した，オーバーサンプリングという技術を使うのです。

図11を見てください。

滑らかな線が実際の波形です。16kHzではサンプリングが実線の通りになります。この棒グラフのような実線のごつごつと，滑らかな線とのギャップが，量子化ノイズです。これを，デジタル的に4倍の周波数で直線補間した場合，どうなるでしょう？その結果は波線の棒グラフのような波形になり，ギャップはかなり減ります。

こうした場合，サンプリング周波数は，16kHz×4＝64kHzといえます。したがって量子化ノイズは16kHz付近から，64kHz付近にシフトされることになりますよね。

実際には8kHzまでの音しか録音されていないので，8kHzで-3dBになり，48kHzで-100dBになればよいことになります。

この場合4倍なのでたいしたことはありませんが，これでも8kHzから16kHzへの急降下より，8kHzから64kHzへの降下のほうが穏やかになることは間違いないはずです。降下が穏やかになるということは，アナログフィルタでの段数が少なくなるということなので，アナログ回路の大きさが小さくなり，ノイズの乗りにくくなって，波形が滑らかになるという，いいことずくめの結果に終わります。

私が今作成しているPCMボードは16倍

か8倍のオーバーサンプリングをする予定です。ただ，こういったことをするとはいえ，やはり，コンピュータに載せるわけですから，最終的には16ビットのうち下位4ビットはノイズの巣窟になると思われます。

まぁこれは，あくまでも余談ということで。

---

*4 もはや献身という。
*5 フィルタは段数が増えるほど，周波数特性の高音域での「落ち」が鋭くなります。

コントローラをどうするか？

フィルタを変えたおかげで31kHzのときのアナログ回路はばっちり決まりました。しかし，このまま16MHzを加えただけでは，従来のモードがなくなっているので，どんなソフトを立ち上げても，音が1オクターブ上がってしまいます。

互換性を重視するには，16/8/4MHzを可変にすべきです。これらを物理スイッチで切り換えるのは，いささか間抜けですから，ソフトウェアでデータセレクタを制御するなんらかの信号を持ってこなくてはなりません。

実際には，どのように16/8/4MHzを変えるか悩んでしまうところです。単純に私が考えたところ，次の2つを思いつきました。しかし，どちらにも利点，欠点があります。

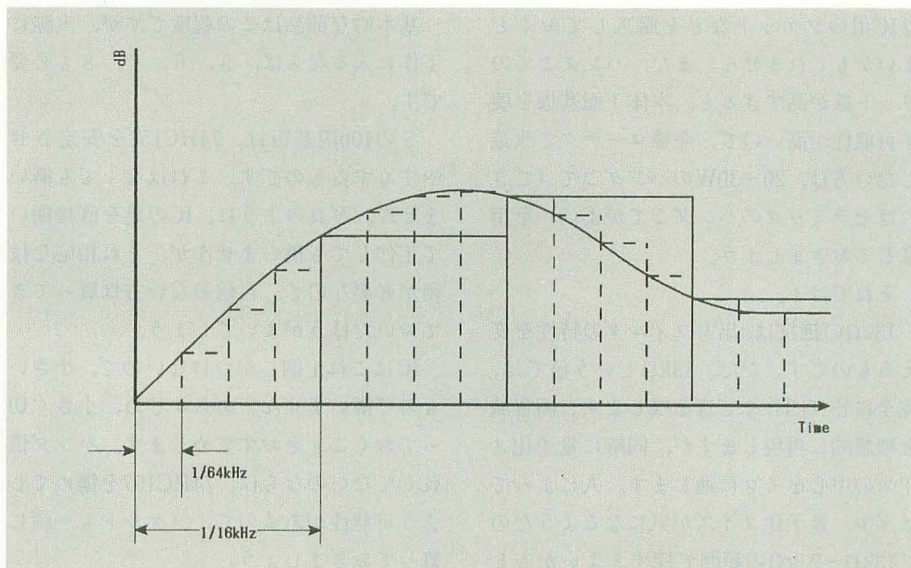
1) ジョイスティックポートの2P，8番ピンを利用する

利点：従来のPCMは完全に網羅するため，PCMに関して互換性が高い。すべて下面基板のみで処理できるなど。

欠点：ジョイスティックポート2にサイバースティックなどの，インテリジェントタイプのジョイスティックが使えない。MDのジョイパッドもだめ。CPSファイターもだめ。広域を網羅するため，どうしても31.2kHzモードにアナログフィルタをあわせると，3.9kHzでは量子化ノイズバリバリになってしまう。

2) 4MHzモードの代わりに16MHzモードを入れる

利点：基板は完全に下面に格納できる。従来の5.1kHzモードが20.8kHzモードになるため，PCM8を常駐しなければ，Z-MUSICが無改造でも楽しめる。さらに

図11 4倍オーバーサンプリング

PCM8を常駐すれば，5.1kHz，3.9kHzのモードも楽しめる。

欠点：4MHzモード，すなわち，3.9kHz，5.1kHzモードがハードウェア的になくなってしまうため，従来のソフトで使っているものは動いても変な音になる。

どちらにもいえる欠点は改造したマシンかしていないマシンか，ソフトでわからない点です。商品じゃなし，ローテクですから，このあたりは環境変数にでも各自設定してもらうことにします。

1)か2)を選択する場合，やはり，ジョイスティックが潰れるということから，1)はいただけません。過去，3.9kHz，5.1kHzを使っているソフトは滅多にありませんでしたし，いま使っている人がいるとは思えません。たとえいたとしても，PCM8を常駐さえしていれば，問題なく使うことができます（逆にソフトウェアオーバーサンプリングしているかもしれない）。

やっぱりジョイスティックは重要だ。ということで，2)の方法をとりました。

これで，従来のよく使われるモードを残しつつ，新しい31.2kHz，20.8kHzが楽しめるようになりました。

改造に関して

原理さえわかってしまえば，あとは簡単です。ただ，実際に回路があるマシンでないと，なかなかコンデンサを発見できないのが難点です。

とりあえず，EXPERT～XVI，030，初期型は回路図が入手できましたので，改造の手順は詳しく書きます。その他のユーザーの人は，図1，図6の回路図でも見て探してください。

編集部にすべてのマシンがあるとは限りませんし，あっても改造してよいマシンとは限らないので，チェックできません*6。申し訳ないですが，それ以外のユーザーは頑張ってほしいところです。

まず部品表です。表1を見てください。購入するとき，部品の数ぴったりだと壊れたときに怖いので，少し多めに買ってきましょう。秋葉原では全部で150円ほどで買え

表1 部品表

	部品	数量
1.	13kΩ(1/4W)	3個
2.	820pF	1個
3.	4.7μF/50V	2個
4.	74HC157	1個
	合計金額はたぶん150円ぐらい	
	あると便利なもの	
5.	100円基板	1枚
6.	ソルダーウィック（ハンダ吸い取り線です。ないとハマるでしょう）	1巻き
7.	これらを接続するコード	
8.	基板を固定する接着剤	

注意）EXPERT～XVIは普通のサイズの部品ですが，030はチップコンデンサ，チップ抵抗です。030基板の改造は実際にはこちらでもまだ行っていないので，一度部品を開け，取り替える部品がどのような部品かチェックした後，購入してくる必要があります。

もっとも，それほど高いわけではないと思うので，両方買ってきてもよいとは思いますが。

る部品ですが，ハンダに自信がない人などはIC用のソケットなどを購入しておくとよいかもしれません。また，ハンダごてのワット数が高すぎると，本体下面基板を壊す可能性が高いので，今後ローテクで改造したい方は，20～30Wのハンダごて（できればセラミックのハンダごてがよい）を用意しておきましょう。

それでは1。

13kΩの抵抗は，出力フィルタの特性を変えるものです。ただ，13kΩという値では，完全に15.6kHzまで音を残します。高音域を理想的に再現しますが，同時に量子化ノイズの中心をモロに通します。人によっては多少，量子化ノイズが気になるようなので13kΩ～20kΩの範囲で探すとよいかもしれません。

私個人の感覚だと，13kΩでは，耳が疲れてしまいます。このあたりを個々で変えると互換性云々といわれそうですが，同じようなことが，インナーホンで聞く/ヘッドホンで聞く/スピーカーで聞く，といった違いで表れてしまいます。アンプのTONEをいじるだけでも変わります。そういうわけで，これは個人の好きにしましょう。

ちなみに，私は気張って金属被膜抵抗を買ってみましたが，普通のカーボン抵抗と，どう音が違うのか，まったくわかりませんでした。ちょっと意味がなかったですね。

2の820pF（表示は821が多い）のコンデンサは，極性のない，セラミックコンデンサ，積層コンデンサ，マイラコンデンサなどを購入してください。ちょっとコンデンサは勉強不足で詳しくないので，具体的にこういうところに利用するとき，どれがいちばんよいか教えていただきたいところです。

これは，入力フィルタを高周波に対応するための部品です。ただし，これだけでは発振するので，3の，4.7μFの電解コンデンサを入れます。これは，入力の低音を倍増するための改造です。

3の4.7μFのコンデンサは電解コンデンサです。本体内で利用していたものは耐圧が50Vと大きいものでした。これらは2カ所変えるので，2つ必要です。

4はデータセレクタです。この場合では，コントローラですね。

74HC157なんてのは，地方のパーツセンターにも売っていると思われます。

基本的な部品はこの程度ですが，実際に工作に入るならば，5，6，7，8も必要です。

5の100円基板は，74HC157を安定させやすくするものです。これはなくても構いません。写真のように，ICの足を直接開いて工作しても構いませんが，それ相応な技術が必要なので，自信のない方は買ってきておいたほうがよいでしょう。

ICはこれ1個しかつけないので，小さいもので構いません。あらかじめ，小さく切っておくことをおすすめします。ハンダ慣れしてないのならば，74HC157を傷めてしまう可能性があるので，ソケットも一緒に買っておきましょう。

6のソルダーヴィックとはいわゆる，ハンダ吸い取り線です。本体についている部品をはずすので，必ずなくてはいけません。大きなハンダ吸い取り器を持っている人は，それでよいと思いますけど。

7のこれらを接続するコードはいうまでもなく必須ですね。8の接着剤は，あれば事故が防ぎやすいということで。両面テープの薄目のやつ（なかに気泡がないやつ）で貼りつけても構いません。

それから，テスターがないと，なにか不都合があったときに修理ができません。ローテクニシャン（？）はメーカーに修理なんて考えてはいけませんよ。持ってない方はいまのうちに買っておくのがよいと思われます[*7]。

*6 もし，ACE，PRO系，Compact系ユーザーの方で改造に成功した人がいたら，報告お願いします。部品番号などをネットワークで流したりすれば，ほかの人がわかりやすいと思います。もちろん，編集部に知らせてくれればうれしい限りです。
*7 6月号のメガディスプレイで，Oh!Xも改造に走るのか！ という葉書が何枚か来ましたが，1年ほど前のローテク特集で，すでに本体改造をやっていたりします。LED変えるのも立派な改造でしょ？ メーカーはネジを開けただけで，修理してくれませんよ（普通）。

本体を剝く

改造するに当たって，最初にすることは，基板を剝き出しにすることです。

まずは本体の蓋を開けることから始めます。一度，1993年7月号のローテク特集で，それなりに詳しく話しましたから，ここではざっと説明することにします。

まずはX68000の電源を切り，コンセントから抜きます。そして，マンハッタンシェイプの背面の黒いネジを，右のタワーからも左のタワーからも同じく3つずつはずします。

そして，サイドを押すようにして，ケースをはずします。これらは試行錯誤でやってください。右のタワーは割合簡単にはずれますが，左のタワーは，フロッピーディスクがあるのでなかなかはずれません。気合を入れてサイドを押し，はずしてください。

次に，X68000の後ろ側の下のほうの部分に下面基板を接続するネジがあります。右のタワーにはRS-232C端子の上の方にひとつ，左のタワーにはジョイスティックポート2の上にひとつずつあります。右のタワーのネジは3mmのネジ，左のタワーのネジはプラスティックに止めるため，木ネジです。組み立てるとき，この2つのネジは，「要」になるネジなので，必ずつけましょうね。

これをはずしたら，X68000を上下逆にひっくり返します。HDD内蔵モデルでも動作してなければ，それなりの対Gがあるので壊れるとは思えません（ただし，ACEは別っぽい）。

とりあえず，下面に見える5つの黒い木ネジをはずします。左側のタワーからは，大小4つのコネクタが，右側のタワーからはひとつのコネクタが接続されているはずです。左のタワー（FDDがあるほう）を接続するコネクタのうち，電源を流すコネクタ（カラフルな太い線がついているコネクタです）は，基板そのものをはずす過程ではずしたほうが賢明です。いまとれなくても構いません。

これで下面基板がはずれます。

はずれたら基板上部から見えるネジをはずします。RS-232C端子を固定するために少々長いネジが2つ。真ん中の少し前よりに，黒いネジがひとつあるはずです。

これらをはずしたら，基板を剝き出しにしてください。多分，ほとんどの場合，「猛烈」にホコリっぽいでしょうから，ハケなどで掃除でもしてあげれば，愛機も喜ぶことでしょう[*8]。

＊8 暖かい分，ゴキブリなどが巣を作っていることがあるそうですよ……想像すると「超」気持ち悪そうですが。あなたの愛機は大丈夫ですか？

基板を加工する

まず先に，コントロ一ラ部を作りましょう。ICはひとつですからそんなに難しい改造ではないですよね。ローテク入門用ってところですか？

回路図は図12です。

書いてはいませんが，電源は5VでこのICの場合は16ピン，グランドは8ピンですから忘れずに。電源は近くの74系ICから取ってきています。初心者の方で「どーしても！」改造したい方は3択あります。

まず，友人のなかから，ハードに強そうな人を探してやってもらう。もちろん責任は頼んだあなたが取るべきでしょう。無理して自分でやる必要もないので，確実な方法かもしれませんが，あなたはこれを選んだ時点で，ハイリスクノーリターンなローテクニシャンの道を放棄したことになります。

次に写真を見ながらやる。

間違いなく動く基板の写真を取ったので，そのまま「完全に同じに」作ればできるはずです。しかし，写真では74HC157が見づらいでしょうから，リスクはつきまといます。あまりおすすめできる方法ではありません。

最後に，自分で書店に行き簡単にわかりそうなハードウェアの本を買ってきて勉強することです。私は向上心がある人は大好きです。そういう人には惜しみなく，協力してあげます。自分にぴったりあう本は人それぞれですから，まず自分がわかる範囲で74系ICの使い方が掲載された本を探すことから勉強が始まります。

頑張ってください。

さて，回路は簡単ですから，できたことにしましょう。できあがった基板を取り付ける前に本体基板にも加工をせねばなりません。

以後，特に初期型，EXPERT～XVI，030を中心に話を進めます。実際に改造をしたのは，XVI，EXPERT1，2です。回路図があれば，部品番号がわかるので，初期型，030の部品番号は，Outside X 68000，X68030Inside/Outから引用しました。

さて，まずソルダーヴィックを用意します。部品をいくつか取り替える際に，現在ついている部品をはずすためです。

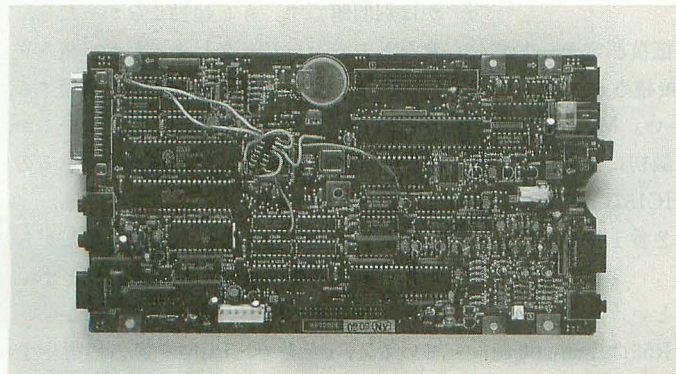
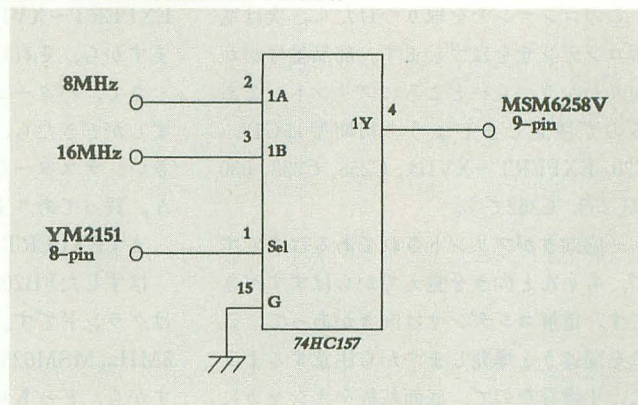
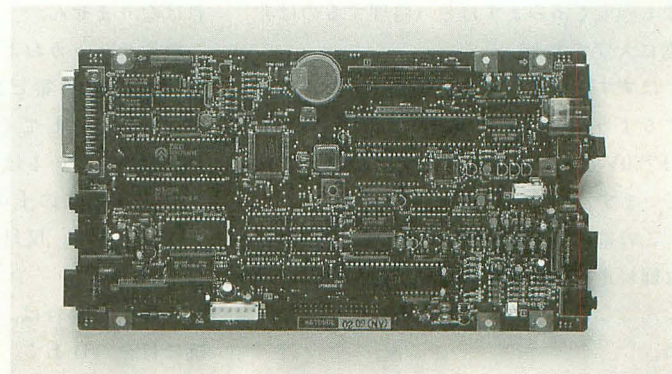
最初は，出力フィルタにある，3つの68kΩの抵抗を探します。初期型はR25，R41，R26，EXPERT～XVIはR232，R233，R234，X68030はR24，R246，R236です。部品番号のプリントは，割とわかりづらいので，各自大きさを必ず確認してから取ること。はずす抵抗の色は青，灰色，だいだい，金色のはずです。030はチップ抵抗だったはずです，ちゃんとチェックするように。

はずすコツは，まず，ソルダーヴィックを暖める部分にしき，上からハンダごてを当てることです。初心者は短気なのか，暖まる前に，まだかな？ とはずしてしまう傾向があります。ちゃんと暖まれば，銅色のソルダーヴィックがハンダを吸い取り始め，銀色になるはずです。初めのうちはなかなか面白いものでしょ？

まわりには，細かいパターンが走っているので，器用に吸い取ってください。これでほとんど取れますが，一見ハンダがないようでも，微弱なハンダで，抵抗はつながっているものです。はずす抵抗をニッパーなどであらかじめ切っておくほうが，失敗せずに綺麗にできるはずです（元に戻したければ，68kΩを買ってくればよいのですから）うまく切ったら，足の先をピンセットなどでつまみ，反対側からゆるめます。ゆるくなってきたら，そっと引き抜いてください。こうすれば誰でも取れるでしょう。

綺麗にはずしたら，この部分に抵抗をつけます。13kΩ～20kΩがよいでしょう。大きいほど高音の伸びがなくなる分，ノイズが気にならなくなります。13kΩ以下は無意味です。これ以上はノイズが乗るだけですから。

これだけで，出力フィルタの特性は変わります。ここらで動作実験ということで，少々遊んでみても構いません。ただ，初心者は下手に剥き出しのまま動作させると壊すことがあります。X680x0の動作チェックのポイントは電源を入れて，すぐ，電源を切り，ちゃんと，緑ランプが点滅するか見

図12 PCM高音質化のメイン回路

フィルタまわりを変更

コントローラを取りつける

ることです。これでちゃんと電源が切れたら，どこも接触せずにとりあえず動いていることがわかります。もし，ランプがいきなり消えてしまうのに電源が落ちていなかった。点滅もしない。そういう場合は，即座に後部メインスイッチを切りましょう(PRO/Compactはすぐにコンセントを抜くように)。

次は入力フィルタです。

初期型はC31，EXPERT～XVIまではC257, 030はC261です。これはさっきの抵抗の近くにあるはずです。もともとついているコンデンサは1800pFですから，多分，182と書かれているでしょう。

はずすコツは，さっきの抵抗と同じようにすることです。もとに戻したくなったら，同じように1800pFを買ってくればよいのですから，工作に自信がない方は最初にこのコンデンサの番号を確認し，間違いないことがわかってから，足を切り，はずしてください。まわりの細いパターンは切れると直すのが面倒なので，注意してはずしてくださいね。

取れたあとは820pF (821) のコンデンサを取りつけます。抵抗と同じく方向性はありませんから，適当に向きを揃えて綺麗につけてください。

このコンデンサを取りつけたら，次は電解コンデンサをはずします。部品番号がかなりわかりづらいところにプリントしてあるので注意しましょう。初期型はC18, C20, EXPERT～XVIは，C258, C233, 030はC259，C262です。

一応向きがプリントされてあるはずですが，ちゃんと向きを覚えてからはずすべきです。電解コンデンサは向きがあって，これを間違うと爆発しますから注意するように。小爆発なので，底面基板をオシャカにする程度で済みますけど（修理するのは非常に大変ですよ)。

はずすときは，大きさを間違いなく確認するように。はずす電解コンデンサは1μF/50Vです。直接大きさがプリントされています。

この電解コンデンサのまわりにも，重要な線が走っていますので（重要じゃない線なんてないと思うけど）注意してくださいね。

綺麗にはずれたら，方向を確実にあわせて，4.7μF/耐圧50Vの電解コンデンサをつけてください。

これでフィルタ部分は完成します。

さて，ここまでできたら，次はMSM6258に関する部分です。

初期型は回路図しかないのでわかりませんが，どうもMSM6258そのものが違うようです。EXPERT以降は，9番ピンがクロック入力なのに，初期型では13番ピンにクロック入力があります。この13番ピンは，IC33のLS00の8番ピンから出ているので，うまいところでパターンカットをする必要があります。パターンカットは，まわりを傷つけないように，かつ，狙ったパターンは確実にカットしてください。テスターなどで，パターンがちゃんとカットされているか，隣のパターンはちゃんと接続されているか，常時チェックすべきでしょう。

EXPERT～XVIまではFB202（3本足のコンデンサみたいなやつ）をはずせば片はつきます。パターンカットは必要ありません。はずす方法も前と同じく，最初に部品の足を切ったほうがよいでしょう。足が多い分，取りはずしが大変ですが，3本ぐらいなら簡単なほうです。頑張ってください。

回路図しか見ていませんが，030もFB210をはずせばよいようです。おそらくこれもEXPERT～XVIと同じように取りはずせますから，それほど難しくはないでしょう。

さて，パターンカット，および，取りはずしができたら，テスターを用意してください。テスターなしにローテクはできません。買ってありますよね？

まずEXPERT～XVI，および030の場合。

はずしたFB202および，FB210の真ん中はグランドです。右側，左側のいずれかが，8MHz, MSM6258のクロック入力になりますから，どっちがどっちかチェックしなければなりません。

導通チェッカおよび，抵抗測定モードで，MSM6258の9番ピンと接続されているほうをチェックしてください。MSM6258側に接続されているほうは最初に作ったコントローラ回路の上の74HC157の4番ピンと接続します。反対側を2番と接続してください。

初期型の場合は，R55という4.7kΩの抵抗に目をつけます。このR55は片側は電源，片側はMSM6258と接続されているので，MSM6258の13番ピンと接続されているほうから線を伸ばし，コントローラ回路の74HC157の4番ピンに接続します。一方，IC33, 74LS00の8番ピンは8MHzが出力されているはずなので，ここからコントローラ回路の74HC157の2番ピンに接続します。

ここまで終わったら，本体基板から16MHzが出ている部分を探します。EXPERT～XVIまでは16MHzのオシレータの5番ピン（右肩）の近くの太いパターンが16MHzですから(テスターで各自確認するように)，ここから16MHzをとり，コントローラ回路の74HC157の3番ピンに接続します。

最後にこの74HC157の1番ピンですが，これは本体基板上のYM2151の8番ピンから引っ張ってきます。

74HC157の電源は適当な74系ICから引っ張ってきてください。また，Gの15番ピンはそのままグランドに落とせばよいので，これも適当な74系ICから奪ってきてください。

以上，すべてにおいて，EXPERT～XVIは実際に作業を行ったうえでの注意を，初期型，030は回路図を片手に，EXPERT～XVIの改造をもとに机上で考えています。

よって，030，初期型ユーザーは改造の際，『Outside X 68000』，『X 68030 Inside/Out』を片手に自分自身でも確認しながら，改造を行ってください。

例によって，責任を取るのは，改造を決意した本人なのですから。

動作確認

とりあえず動作確認をしましょう。

今回の改造の場合，ちゃんと起動できるかは別問題です。もし起動しなかったら，変なところをいじってしまったか，組み立てに失敗しているか，基板同士を接続する，5つの「コネクタ」を接続し忘れているか，どれかです。特にコネクタは結構忘れるのでご注意。

無事に起動したら，とりあえずPCMがちゃんと動くかチェックしないといけません。いちばん簡単なのは，Z-MUSIC本についているZVTなどで，PCMを適当に再生することです。一応PCM8は常駐解除してく

ださい（OFFじゃだめです）。

まず，適当なPCMをロードして，PCMをそのまま再生してください。出力フィルタを変えたおかげで，こもった感じがしなくなったはずです。ハイハットなどは，如実に違いがわかるでしょう。

次に周波数を変えてみます。3.9kHzモードでも15.6kHzで再生されるでしょう。5.2kHzにしたら音が高くなるはずです。

ここまでできたら安心です。改造は成功しました。31.2kHzはプログラムがそれに対応しない限り再生できないので，とりあえず20.8kHzモードでお茶を濁しておいてください。

ノイズが気になる方は，抵抗13kΩを20kΩぐらいにすると改善されます。ただ，正面ヘッドホン端子の場合，強烈にノイズが乗るので，外部AUX出力からアンプなどに通して再生してみてください。

さて今度は入力です。

とりあえずZVTのモニタモードでPCMをサンプリングします。CDなどをモニタモードから直接サンプリングしてみれば音の違いがわかるはずです。再生しながら，周波数を7.8k，10.4k，15.6k，5.2kHz(20.8kHzモードになる）と変えてみれば，いろいろと楽しいでしょう。高音の伸びがどう変わっていくか，試してみてください。

使用感

ダイナミックレンジに関してはMSM6258を利用する限りなんともできないのが残念です。しかし，高域に関しては伸びがよくなり，クリアになると思われます。

現在のままでは，ZVTで遊んでも，31.2kHzモードがお聞かせできないので残念です。ZVTに関しては，多少パッチを当てれば済みそうなので，善ちゃんに頼んで，後日，パッチを当ててもらうことにします。

Z-MUSICはPCM8が対応すれば，すぐ対応するそうです。通信で出回っているソフトにRDNというPCM8互換音程変化対応ソフトがあるそうで，こちらのほうでも対応してくれればうれしい限りです。

しかし，PCM8に関していえば，データ量2倍になる分，単純に2倍ぐらいの時間がかかるため，10MHzでも重かった処理がどうなってしまうのかわかりません。

無改造XVIでも問題が出てきてしまう曲があるかもしれません。

そろそろ，石上さんのアクセラレータができた時期なので，それに期待！　ってことになるんでしょうか？

いい忘れたことを最後にまとめておきましょう。まず，MSM6258にとって16MHzはオーバースペックです。熱を持つかもしれません。MSM6258は冷やして使うことをおすすめします。

個体差によって16MHzが動かないかもしれません。そのときは各自考えてください。

技術資料

原理をほとんど説明してしまったので，いまさら技術資料もないですが，対応ソフトウェアを作ってくれる人のために，ある程度，ここでまとめておきましょう。

本改造では改造されたハードとノーマルハードとをソフトウェアで判断する方法がありません。したがって，改造マシンにソフトを対応させるならば，いろいろとしなくてはならないことが出てきます。

ソフトウェアから見たとき，今回の改造では，4MHzモードを16MHzモードにしただけです。8MHzは従来のまま生きているので，ソフトは4MHzモードがどのように変化したか知る必要があります。

これを環境変数でごまかすことにします。

本改造をした場合，OS上で環境変数に，

set OPM_CT1=16

をセットするようにしてください。

ソフトウェアでこの改造に対応する場合，このOPM_CT1から数字を読み取り，対応させてください。対応させる場合，16MHz以外の周波数を入れているユーザーが出てくることも考え，できる限り，数値的に小数点以下まで読み取ってください。値は10の3乗倍すること。

31.2kHzモードというのは，この周波数の512分周，20.8kHzモードというのは768分周です。ただ，高速な処理をしている場合，周波数変調をする時間はないでしょうから環境変数がセットされているか，セットされていないかを見るだけでいいかもしれません。

互換性を含めて，改造は16MHzを入れることをおすすめします。

改造を施した場合，PCMの出力レートは，表2のようになります。

8255ポートCの設定方法，CTレジスタの設定方法は，この場では省略します。Inside X68000を参照してください。

現時点において，多機能再生プレイヤーはすでにできています。いろいろな付加機能をつけたせいで，リストが長くなってしまい，残念ながら，今回はリストを掲載することができません。周波数変調ライブラリ，高出力ライブラリを添付して，後日のオマケディスクのときに収録予定です。

なお，Inside X68000の298ページの図38はCT2，CT1の表示が逆です。機能はそのままですが。正解は268ページの図7，$1Bの項を参照してください。

まとめ

今回は大胆に本体改造に踏み切ってみました。

こういう改造になると，自分の持っているパソコンの回路図，仕様書は必ず必要です。ローテクニシャンにとって，74規格表，OUTSIDEは必需品ということですか。

今回の回路も前回の回路と同様に責任は取れません。これは毎月いっていることですけど。

しかし，やってみたけど，どうしても動かない，自分ではわからないけど，編集部の誰かは知っているかもしれない。そういう藁にもすがりたい気持ちはよくわかります（改造に失敗した場合など）。

まず，自分で調査をし，おかしくなってもすぐ諦めず，自分が実際にいじった部分を確認して修理に挑戦してください。

生きた情報がほしいなら，PCVANのX1CLUBのフォーラム7ハードウェアにて，ある程度サポートできます。私よりハードの凄腕がいるので，ぜひいらしてください。

表2　出力PCMレート

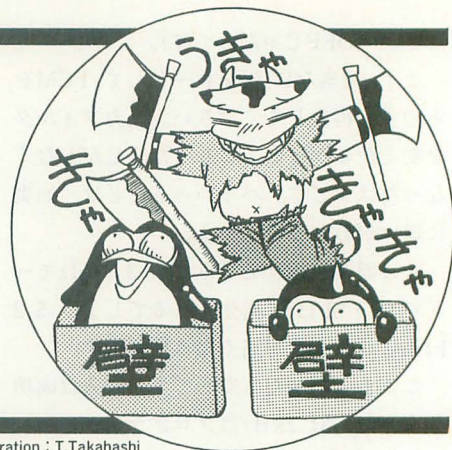
CT1	8255C		出力周波数
	3	2	
0	0	0	8MHz/1024= 7.813kHz
0	0	1	8MHz/ 768=10.417kHz
0	1	0	8MHz/ 512=15.625kHz
1	0	1	16MHz/ 768=20.833kHz
1	1	0	16MHz/ 512=31.250kHz

注意）CT1はOPMレジスタのCT1です。
8255Cは8255のCchです。

(で)のショートプロぱーてぃ――その59

3発殴って殴って～

Komura Satoshi 古村　聡

見た目からは想像できない(で)氏の得意なゲーム。今月のプログラムには，そのジャンルのゲームが登場しますが，(で)氏は攻略できたのでしょうか？　ぷろぐらむ風まかせでは，プログラムに渡す引数を３種類の言語について解説しています。

illustration：T.Takahashi

さて，私の得意なゲームとはなんでしょう。ふっふっふ。

ソニックブラストマンっていうゲームマシンにあるではないですか。そう，「３発殴っていん石を壊せ！」とか表示されて，パンチングボールを殴って，パンチ力が何kgかを競う力自慢機。デパートの屋上のゲームコーナーに置いてあったりしますよね。でも，私の得意はあれではなくて，敏捷性を競うもぐらたたきタイプのゲームなのです。どうだ，イメージと違うでしょ。うふふふ。

特に，ナムコの「ワニワニぱにっく」でたくさん叩いた証拠の「まいった！」をわりとよく出していたのが自慢だったんですよ。

ところがですね。「ワニワニぱにっく」の後継機の「カニカニぱにっく」。要するにもぐらたたきを横アナからカニが出てくるようにしただけなんだけど，こいつがワニワニに比べると難易度が妙に高いんです。カニがちょろっと出てきたと思ったら引っ込む，って非常に精神衛生上悪い動きをするんです，これが。

初めてやった私は１度やって非常にムッとしてしまって……２度目はさらに点が悪くなって……ついに３度目のゲームで，

「うりゃぁあああっ！」

とばかりに思いきり叩いたら……ガツーンと，ハンマーの柄の部分にカニが当たってポ～ンと空高く飛んでいってしまったんですねぇ。

おかげで私は「３回殴ってカニを壊した男」と呼ばれてしまったのでありました……。あのな。

AX68K.X

何回でも敵を殴れ！

さてさて，それでは最初のプログラムにいきましょう。もぐらたたきなんですね，これが。では３回殴って……(もう，いいって)AX68K.Xです。どうぞっ。

AX68K.X for X680x0
(要C Compiler PRO-68K)
千葉県　森川忠敬

壁から出てくる敵を斧で殴るのだ！　このプログラムはX-BASICのリストの形で書かれていますが，コンパイラ専用です。「C Compiler PRO-68K」でコンパイルしないと遊べないので気をつけてください。ED.XやX-BASICのエディット画面でリスト１を入力してから「AX68K.BAS」という名前でセーブしてください。

それから，このプログラムはウェイト時間の調整のためにXVI.Oというファイルが必要になります。ソースリストのXVI.Sが本誌1992年11月号に掲載されていますのでそれを用意してください。XVI.Sのリストをエディタなどで打ち込み，セーブしてから，

A>AS XVI.S

とアセンブルすればXVI.Oを作れます。

AX68K.BASとXVI.Oがそろったら，

A>CC AX68K.BAS XVI.O

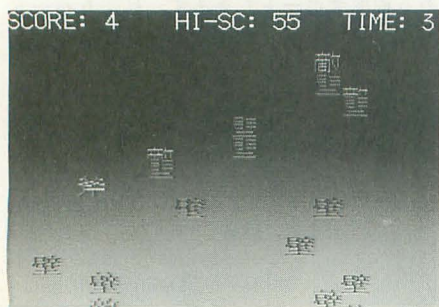
としてコンパイルしてください。.BASなどの拡張子も省略しないで全部入れてくださいね。

さて，無事打ち込み終わったら今度は遊び方。このゲームはマウスで操作します。画面のそこかしこの「壁」という表示の裏から「敵」が出てきます(本当に「敵」って表示なんだ，これが)からマウスで斧を敵のところまで持っていってください。マウスカーソルが敵の上にいったところで敵は殴られたことになります。マウスをクリックする必要はありません。

敵のスピードは３段階で，敵を殴るとそれに応じた得点が入ります。制限時間は３０秒。タイトルが出ているときに［ESC］キーを押すと終了します。

投稿原稿には，壁から敵が出てきますって書いてあるからどんな風になっているんだろうと思ったら，本当に漢字で「壁」「敵」って書いてあるし。ん～，なんかシュールですよねぇ。しかも，この敵が，ただの文字のくせに壁から頭(？)を出す動きが妙になめらかで，本当に敵が頭を出しているような気になるから不思議ですねぇ。ああ，表意文字文化であることよ。

これって，結構難しいですね。いつもSX-WINDOWなんかでなにげなくマウスを使っているんだけど，なかなか思ったようにいかない。いや～，何度力まかせにマウスを殴ろうかと思ったことか。さすがに「マウスすらも撲殺した男」というあだ名がつくのは避けたいのでやめましたけど。

しかし……どうしてこう墓穴を掘るの，わしってば。

夜型人間に人権を！

朝起きて，学校や会社に行くとき，テレビをつけておくと，画面にいつも時刻が表示されていて，時計よりも重宝してしまいます。でも，テレビに時刻が表示されているのって朝とか夕方の決まった時間だけですよね。それじゃ，不規則な生活をしている編集さんみたいな人はかわいそうです。だから，いつも時刻が表示されていればいいと思いません？　そんなときに使えるのがこのCLK.Xなのです。では，どうぞっ。

CLK.X for X680x0

(要アセンブラ，リンカ，ディスプレイテレビ，テレビコントロールケーブル)

東京都　北見　英一

このプログラムはアセンブラのソースリストの形で書かれています。まずはED.Xなどのエディタでソースリストを打ち込んでください。ED.Xの場合は，

A>ED CLK.S

でエディタを立ち上げ，入力します。

[ESC] E

で，ファイルを保存してエディタを終了します。

それから

A>AS CLK.S
A>LK CLK.O

でアセンブル，リンクを終えたら作業は終り。CLK.Xができていますね。

あとはコマンドライン上からであれば，

A>CLK

とすれば，時刻が表示されます。テレビ兼用のディスプレイで本体とディスプレイの間がテレビコントロールケーブルでつながっていれば，テレビ放送画面にX68000の表示する現在の時刻が重なって表示されます。

私のX68000もテレビ付き(いまや懐かしいグレーでモノラル音声なディスプレイテレビ)なので使ってみましたけど……。グッドグッド。

このプログラムはX68000のスーパーインポーズ機能を使っているわけですが，スーパーインポーズってご存じですか？　そう，要するに画面の字幕みたいなものなんですが，X68000ではスーパーインポーズモードにすると画面に黒で描いてある部分を透明にして，そこにテレビの画像を張り込むことができるようになっているのです。このスーパーインポーズ機能を知っていればみんな送ってくるだろうと思ったタイプのプログラムだったんですが，いままで送られてこなかったってことは，もしかしてスーパーインポーズって機能自体知られてないんだろうか……んなわけないよね。

そうそう，このプログラムではV-DISP割り込みが使われています。ディスプレイの表示って，X68000が上から順に一定の時間ごとに描き直しているんですが，上から下まで描き終わると次の描くタイミングまで画面の表示をしないんですね。非常に高速(だいたい１間隔が1/60秒くらい)なので目には見えないですけど。

で，実際の使い方なんですが，投稿原稿によると，

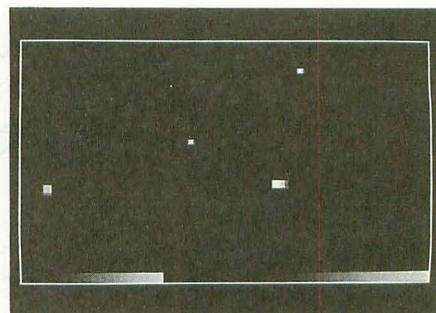
「V-DISP割り込みについて」

このプログラムの動作を簡単に説明すると，V-DISP割り込みで約０.５秒おきに割り込みをかけて，フラグ１をリセット($FFにする)します。メインプログラムではフラグ１が$FFになるまで待って，$FFになったらフラグ１をクリアしてから時刻の表示をします。時刻を表示する部分は，IOCSコールで本体の時刻をBCD形式で受け取り，それを自前のフォントデータを利用して表示します。

V-DISP割り込みは，普通IOCSコールのVDISPSTで設定しますが，常駐プログラムなどで割り込みが使われているとVDISPSTで設定ができないので，一度割り込み禁止にして設定しなければいけません。しかし，この方法だとプログラム終了時に割り込みを完全に(カウンタの設定を)戻すことができません。

そのため，MFP(マルチファンクションペリフェラル)と割り込みベクタを直接アクセスして保存してから新しい割り込みを設定しています。プログラム終了時は先ほど保存したデータを元の場所に戻してから，OSに戻ります。

なのだそうであります。

ただ，この方法でも完全にカウンタの設定を戻したとはいえません。なぜなら保存した値が最大値になってしまうからです。うまく最大値のときに保存できれば問題はないのですが，そうとは限りません。では最大値を保存しておくにはどうするのか？　これは宿題にしておきましょう。

とりあえず，手持ちのプログラムをいろいろ使ってみても問題なかったので大丈夫だと思います。

あ，親切な解説をありがとうございました，北見さん。感謝感謝です。はい。

TAISEN.BAS

いけいけホーミングでごぉごぉ！

さっそく，作戦の解説をする。

諸君らは，「つややか弐号」に搭乗し，輸送機体で前線に投下され，そこで敵機体と戦闘を行う。自分の能力を信じ，機体の性能を最大限発揮して，生き残ってほしい。

以上。

またいきなり投稿原稿を引用してしまいました(楽でいいな，このパターン)。ということで今月最後のプログラムはX-BASIC用のショートな対戦ゲームプログラム，TAISEN.BASです。どうぞっ。

TAISEN.BAS for X680x0

(X-BASIC，要VECTOR.FNC，ジョイスティック２本)

愛知県　水野真也

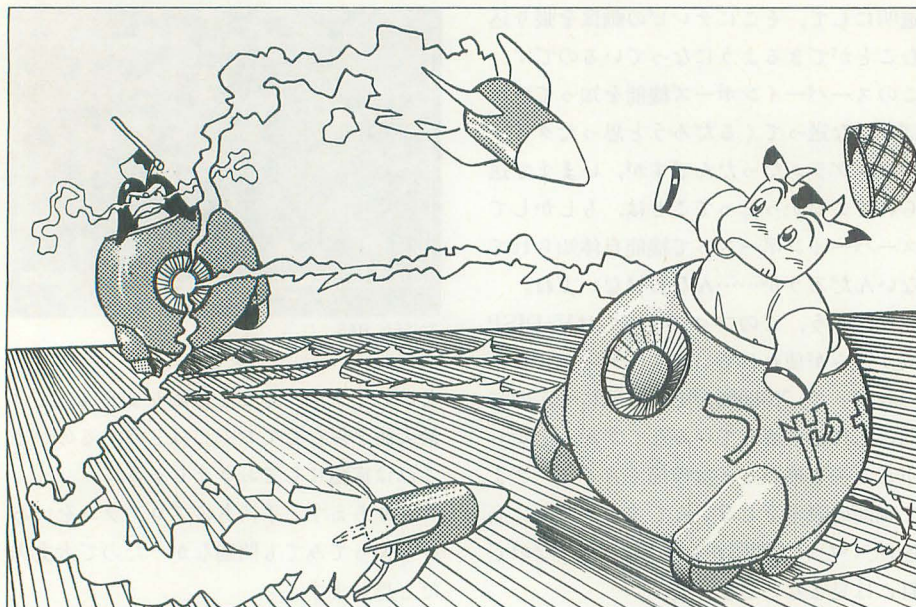
このプログラムを使うにはX-BASIC用の外部関数VECTOR.FNCが必要になります。このVECTOR.FNCは本誌1993年９月号のショートプロぱーてぃに掲載されています。VECTOR.FNCをBASIC.Xと同じディレクトリにコピーして，BASIC.CNFファイルに，

FUNC = VECTOR

と書いてVECTOR.FNCを使えるようにするのを忘れないでくださいね。

そしてBASICを立ち上げて，リストを入力したらRUNでゲームスタート。おっと，そうそう。ジョイスティック２本をつない

でおいてくださいね。

さて，2人のプレイヤーがそれぞれジョイスティックを握りまして，自機を操って敵を倒してください。ジョイスティックで，自機を回転させ，Aボタンでミサイル発射，Bボタンがアクセル。アクセルによる加速は，チャージ(下のゲージ)がある分だけできます。ミサイルは，ある程度ホーミングします。でもって敵にミサイルを当てれば勝利！　なんでありますね。

こ，このミサイルちょっとだけホーミングするんだけど……。コツは敵をひきつけてから撃つことなのかなぁ？　でもあんまり敵が近くにいると今度は自分がやられちゃうし。難しいもんです。

ジョイスティックのBボタンを押すだけでミサイルがそれっぽいところに飛んでいくのはVECTOR.FNCのおかげなんですね。上手に使ってもらえて，ショートプロ冥利につきるってぇもんですね，うんうん。ほかにもX-BASICの外部関数でほしいものってありそうなんで，どんどん送ってくれると嬉しいですね。ついでに「ネットへのアップ自由」になっているとシスオペ冥利にもつきるってもんです(なんか最近私情が多い気が)。

あ，そういや，今月号からショートプロは変わるかもしれない，とか書いてたんでしたっけか？　なんかちっとも変わらなかったような気が……えーとえーと……あはは，大丈夫だよね，みんな先月号のことなんか憶えてるわけないし(オイオイ)。

なんか，自分で墓穴を掘りまくってるよーな気がするのはなぜなんでしょ。穴掘り体質なんでしょうか？　前世はもぐらか炭鉱夫か。いいや。こんなときは布団にもぐって寝ちゃおうっと。それでは来月までお休みなさい。

リスト1　AX68K.BAS

```
 10 screen 0,2,1,1:window(0,0,511,511)
 20 mouse(0):mouse(4):msarea(0,0,241,241)
 30 dim int ex(62),ey(62),co(62),cc(62),fl(62),ra(62),spd(62)
 40 int x,y,bl,br,sc2,sc,tim,en,hsc=50:str t1
 50 if b_argc=1 then en=15 else en=val(b_argv(1)):en=(63 and en)
 60 sprite():sound():title()
 70 /*
 80 while 1
 90   sc=0:tim=31
100   cls:home(0,0,0)
110   repeat:msstat(x,y,bl,br):until bl or inkey$(0)=chr$(27)
120   home(0,256,0)
130   if bl<>-1 then end
140   for i=0 to en-1
150     ex(i)=int(rnd()*16)*16
160     ey(i)=int(rnd()*14)*16+32
170     cc(i)=1
180     sp_set(i+1,ex(i)+16,ey(i)+16,258)
190   next
200   sp_on(0,en)
210   cls:print"SCORE: 0"
220   locate 12,0:print"HI-SC:";hsc
230   locate 24,0:print"TIME: 30"
240   for i=0 to 2
250     sp_set(0,136,128,261-i)
260     vwait(35)
270   next
280   locate 15,6:print" ":sc2=hsc
290   sp_on():setmspos(120,112):t1=time$
300   repeat
310     for i=0 to en-1
320       mspos(x,y)
330       sp_set(0,x+16,y+16,256)
340       if co(i)>spd(i)*10 and abs(x-ex(i))<12 and abs(y-ey(i)
+co(i)/spd(i))<12 then {
350         m_play(spd(i))
360         sc=sc+4-spd(i):co(i)=0:cc(i)=-1:fl(i)=0
370         locate 6,0:print sc
380         if sc>=hsc then hsc=sc:locate 18,0:print hsc
390       }
400     next
410     for i=0 to en-1
420       if fl(i)=1 then {
430         co(i)=co(i)+cc(i)
440         if co(i)=spd(i)*18 then cc(i)=-1
450         if co(i)<0 then cc(i)=1:co(i)=0:fl(i)=0
460       } else {
470         ra(i)=rnd()*1000
480         if ra(i)<5 then fl(i)=1:spd(i)=int(rnd()*3)+1
490       }
500     next
510     vwait(0)
520     for i=0 to en-1
530       sp_set(i+en+1,ex(i)+16,ey(i)-co(i)/spd(i)+16,257)
540     next
550     if t1<>time$ then {  t1=time$:tim=tim-1:locate 24,0
560                          locate 29,0:print tim
570     }
580   until tim=0
590   sp_off():for i=0 to en-1:sp_set(i+en+1,0,0,0):next
600   if sc>sc2 then home(0,0,256) else home(0,256,256)
610   repeat:msstat(x,y,bl,br):until bl
620   msbtn(0,0,0)
630 endwhile
640 /********************
650 func sprite()
660  dim char sp(255),pal(5)={9,5,12,9,9,9}
670  sp_init():sp_clr()
680  sp_on(0,20):sp_disp(1)
690  fill(32,0,47,15,1):sp_color(1,0)
700  for i=0 to 5
710   symbol(i*16,0,mid$("斧敵壁１２３",i*2+1,2),1,1,1,pal(i),0)
720   get(i*16,0,i*16+15,15,sp)
730   sp_def(i,sp,1)
740  next:wipe()
750 endfunc
760 /********************
770 func sound()
780  m_init()
790  for i=1 to 3
800   m_alloc(i,100):m_assign(i,i)
810  next
820  for i=1 to 3
830   m_trk(i,"o"+str$(6-i)+"@66v15l64fedc")
840  next
850 endfunc
860 /********************
870 func title()
880  symbol(8,50,"AX68K",2,1,2,209,0)
890  symbol(10,190,"Push Button To Start",1,1,2,235,0)
900  symbol(10,356,"NEW RECORD!",1,3,2,18,0)
910  symbol(6,352,"NEW RECORD!",1,3,2,248,0)
920  symbol(284,356,"GAME OVER",1,3,2,18,0)
930  symbol(280,352,"GAME OVER",1,3,2,112,0)
940  apage(1)
950  for i=0 to 63
960   palet(i,i)
970   fill(0,i*4,255,i*4+3,i)
980  next
990 endfunc
```

▶テレビの見られるパソコンなどという宣伝をしているが，「それがどうした」といいたくなる。
黒澤　典義(17)群馬県

リスト2 CLK.S

```
  1: *
  2: *        テレビ画面に時刻を表示するプログラム
  3: *
  4: *                  制作：きたみ ひでかず
  5: *
  6: *        Version 1.00     94-03-17    22:35:50
  7: *        Version 1.10     94-03-21    04:13:52
  8: *        Version 1.11     94-03-21    15:30:48
  9: *
 10:
 11:          .include         iocscall.mac
 12:          .include         doscall.mac
 13:
 14:          .text
 15:          .even
 16:
 17: *======================================*
 18: *                  マクロ定義            *
 19: *======================================*
 20: _Store:  macro    dsp
 21:          move.w   (a3)+,dsp*$0200+$0000(a1)
 22:          move.w   (a3)+,dsp*$0200+$0000(a2)
 23:          move.w   (a3)+,dsp*$0200+$0080(a1)
 24:          move.w   (a3)+,dsp*$0200+$0080(a2)
 25:          move.w   (a3)+,dsp*$0200+$0100(a1)
 26:          move.w   (a3)+,dsp*$0200+$0100(a2)
 27:          move.w   (a3)+,dsp*$0200+$0180(a1)
 28:          move.w   (a3)+,dsp*$0200+$0180(a2)
 29:          endm
 30:
 31: *======================================*
 32: *                  プログラム            *
 33: *======================================*
 34:          suba.l   a1,a1                    * Super Visor Mode
 35:          IOCS     _B_SUPER
 36:          move.l   d0,-(sp)
 37:
 38:          move.l   #$0010_ffff,-(sp)        * Save Screen Mode
 39:          DOS      _CONCTRL
 40:          addq.w   #$04,sp
 41:          move.w   d0,-(sp)
 42:
 43:          moveq.l  #$09,d1                  * 512x512 15[KHz]
 44:          IOCS     _CRTMOD
 45:          IOCS     _B_CUROFF
 46:
 47:          moveq.l  #$0f,d1
 48:          IOCS     _TVCTRL
 49:
 50:          move.l   #$0000_0001,$e82200      * テキストパレット設定
 51:          move.l   #$0000_ffff,$e82204
 52:
 53:          bsr      _Data_Make               * フォントデータ作成
 54:
 55:          lea.l    _Data(pc),a0             * 割り込みの情報を保存
 56:          move.l   $0134.w,(a0)+            * Address
 57:          move.b   $e88003,(a0)+            * AER
 58:          move.b   $e88007,(a0)+            * IERA
 59:          move.b   $e88013,(a0)+            * IMRA
 60:          move.b   $e8801f,(a0)+            * TADR
 61:
 62:          move.w   sr,-(sp)                 * 新しい割り込みの設定
 63:          ori.w    #$0700,sr                * Mask Interrupt
 64:          move.l   #_VDISP,$0134.w          * Address
 65:          andi.b   #$ef,$e88003             * AER
 66:          ori.b    #$20,$e88007             * IERA
 67:          ori.b    #$20,$e88013             * IMRA
 68:          move.b   #$1e,$e8801f             * TADR
 69:          move.w   (sp)+,sr
 70:
 71: *--------------------------------------*
 72: *                  Main Loop             *
 73: *--------------------------------------*
 74: _Loop1:
 75:          btst.b   #$01,$800.w              * [ESC]
 76:          bne      _Quit
 77:          move.b   _Flag1(pc),d0
 78:          beq      _Loop1
 79:          clr.b    _Flag1
 80:          bsr      _Time_Display            * Main Routine Call
 81:          bra      _Loop1
 82: _Quit:
 83:          lea.l    _Data(pc),a0             * 割り込みを元に戻す
 84:          move.w   sr,-(sp)
 85:          ori.w    #$0700,sr                * Mask Interrupt
 86:          move.l   (a0)+,$0134.w            * Address
 87:          move.b   (a0)+,$e88003            * AER
 88:          move.b   (a0)+,$e88007            * IERA
 89:          move.b   (a0)+,$e88013            * IMRA
 90:          move.b   (a0)+,$e8801f            * TADR
 91:          move.w   (sp)+,sr
 92:
 93:          moveq.l  #$1d,d1
 94:          IOCS     _TVCTRL
 95:          move.w   #$0010,-(sp)             * 画面を元に戻す
 96:          DOS      _CONCTRL
 97:          addq.w   #$04,sp
 98:          move.l   (sp)+,a1                 * User Mode
 99:          IOCS     _B_SUPER
100:          clr.l    -(sp)
101:          DOS      _KFLUSH
102:          addq.w   #4,sp
103:          DOS      _EXIT
104:
105:
106: *======================================*
107: *                  時間の表示            *
108: *======================================*
109: _Time_Display:
110:          clr.b    _Flag3                   * 0制御フラグリセット
111:          IOCS     _TIMEGET                 * d0.l=$00 HH MM SS
112:          move.b   #$0a,d0
113:          not.b    _Flag2
114:          beq      _Skip1
115:          addq.w   #$01,d0
116: _Skip1:  ror.w    #$04,d0                  * d0.l=$00 HH ?M M0
117:          rol.l    #$08,d0                  * d0.l=$HH ?M M0 00
118:          lea.l    _Font(pc),a0             * フォントアドレス
119:          lea.l    $e01004,a1               * 必ず偶数
120:          lea.l    $e21004,a2               * 必ず偶数
121:          moveq.l  #$04,d1
122: _Loop2:  rol.l    #$04,d0
123:          bsr      _CHR_Display
124:          dbra     d1,_Loop2
125:          rts
126:
127:
128: *======================================*
129: *                  文字表示              *
130: *======================================*
131: _CHR_Display:
132:          move.l   d0,-(sp)
133:          andi.w   #$0f,d0
134:          bne      _Skip2
135:          tst.b    _Flag3                   * 0制御
136:          bne      _Skip2
137:          moveq.l  #$0b,d0
138: _Skip2:
139:          mulu.w   #$60,d0                  * 手抜き
140:          lea.l    $00(a0,d0.w),a3
141:          _Store   0
142:          _Store   1
143:          _Store   2
144:          _Store   3
145:          _Store   4
146:          _Store   5
147:          addq.w   #$02,a1
148:          addq.w   #$02,a2
149:          move.b   #$ff,_Flag3
150:          move.l   (sp)+,d0
151:          rts
152:
153:
154: *======================================*
155: *        V-DISP割り込み                  *
156: *======================================*
157: _VDISP:
158:          move.b   #$ff,_Flag1
159:          rte
160:
161:
162: *======================================*
163: *        フォントデータ作成              *
164: *======================================*
165: _Data_Make:
166:          lea.l    _CHR(pc),a0
167:          lea.l    _Font(pc),a1
168:          moveq.l  #$0a,d3
169: _Loop4:
170:          moveq.l  #$00,d1
171:          move.b   (a0)+,d1                 * Character Code
172:          moveq.l  #$0c,d2                  * 12x24 Dot Font
173:          moveq.l  #$16,d0                  * IOCS   _FNTADR
174:          trap     #$0f
175:          movea.l  d0,a2
176:          moveq.l  #$17,d2
177: _Loop3:
178:          move.w   (a2)+,d0
179:          move.w   d0,d1
180:          lsr.w    #$01,d1
181:          move.w   d1,$0006(a1)
182:          or.w     d1,d0
183:          lsr.w    #$01,d1
184:          or.w     d1,d0
185:          or.w     d0,$0008(a1)
186:          or.w     d0,$0004(a1)
187:          or.w     d0,$0000(a1)
188:          addq.w   #$04,a1
189:          dbra     d2,_Loop3
190:          dbra     d3,_Loop4
191:          rts
192:
193:
194: *======================================*
195: *                  データ領域            *
196: *======================================*
197:          .data
198: _Data:
199:          dcb.b    8,0
200: _Font:
201:          dcb.b    48*24,0
202: _Flag1:
203:          dc.b     0
204: _Flag2:
205:          dc.b     0
206: _Flag3:
207:          dc.b     0
208: _CHR:
209:          dc.b     '0123456789:'
210:
211:
212:          .end
```

▶ 7月号は最後のページにメモ用紙がついていて親切だ(?)。　加藤　恵吾(18)愛知県

リスト3 TAISEN.BAS

```
10 /*
20 /*          ﾀｲｾﾝｳﾁｱｲｹﾞｰﾑ
30 /*                   by -Z- 1994.1.4
40 int x(1,3),y(1,3),d(1),sp(1),en(1)
50 int mx(1),my(1),md(1),msp(1),mch(1),mc(1)
60 int normal_sp=20,missile_sp=45,k=30800
70 int pal(1,19)
80 char game
90 /*          ---===<<< >
100 screen 0,2,0,1
110 console ,,0
120 palet(9,rgb(10,10,31))
130 box(10,10,240,230,9)
140 for a=0 to 19
150   fill(a*4+11,220,a*4+14,229,100+a)
160   fill(a*4+160,220,a*4+163,229,120+a)
170   pal(0,a)=rgb(12+a,0,0)
180   pal(1,a)=rgb(0,12+a,0)
190 next
200 sp_init()
210 sp_disp(1)
220 sprite():cls
230 sp_on()
240 /*          ---===<<< loop >
250 while 1
260 /*
270 for a=0 to 1
280   x(a,0)=100+a*50
290   y(a,0)=125
300   d(a)=a*32+31
310   sp(a)=normal_sp
320   en(a)=0
330   mch(a)=0:mc(a)=0
340   sp_off(30+a)
350 next
360 for a=0 to 39
370   palet(100+a,0)
380 next
390 game=0
400 /*
410 /*          ---===<<< main >
420 repeat
430 /*
440 for a=0 to 1
450   stickin(a)
460   game=game+dddd(a,x(a,0),y(a,0))
470   crt2(a)
480 next
490 /*
500 until game<>0
510 over(game)
520 endwhile
530 end
540 /*
550 /*          ---===<<< funcs >
560 /*                   ---===<<< stickin() >
570 func stickin(a)
580 int st,sg
590   st=stick(a+1)
600   if st=3 or st=6 or st=9 then {
610     d(a)=(d(a)+62) mod 64
620    } else {
630     if st=1 or st=4 or st=7 then {
640       d(a)=(d(a)+2) mod 64
650      }
660    }
670   sg=strig(a+1)
680   if sg=2 then {
690     sp(a)=sp(a)-(en(a)>0)*2
700     en(a)=en(a)-1-(en(a)<0)
710     palet(101+en(a)+a*20,0)
720    } else {
730     en(a)=en(a)+1+(en(a)>18)
740     palet(100+en(a)+a*20,pal(a,en(a)))
750    }
760     sp(a)=sp(a)-1-(sp(a)<normal_sp)
770   if sg=1 and mch(a)=0 then {
780     mx(a)=x(a,0)
790     my(a)=y(a,0)
800     md(a)=d(a)
810     msp(a)=missile_sp
820     mch(a)=2:mc(a)=0
830    }
840 endfunc
850 /*
860 /*                   ---===<<< dddd() >
870 func dddd(a,x1,y1)
880 int hou
890   for i=0 to 1
900     x(a,2-i)=x(a,1-i)
910     y(a,2-i)=y(a,1-i)
920   next
930   x1=x1+vx(sp(a),d(a))/k
940   y1=y1+vy(sp(a),d(a))/k
950   if chack(x1,y1)=0 then {
960     x(a,0)=x1
970     y(a,0)=y1
980    }
990   if mch(a)=1 then {
1000     mx(a)=mx(a)+vx(missile_sp,md(a))/k
1010     my(a)=my(a)+vy(missile_sp,md(a))/k
1011     hou=houkou(mx(a),my(a),x(1-a,0),y(1-a,0),1)
1012     md(a)=houkou2(md(a),hou):mc(a)=mc(a)+1
1013     md(a)=(md(a)+64) mod 64:if mc(a)=50 then mx(a)=0
1020     if chack(mx(a),my(a))=-1 then {
1030       mch(a)=0
1040       sp_off(30+a)
1050      } else {
1060       if hit(a,1-a)=-1 then return(99-a)
1070      }
1080    }
1090 return(0)
1100 endfunc
1110 /*
1120 /*                   ---===<<< crt2() >
1130 func crt2(a)
1140   for i=0 to 2
1150     sp_move(a*10+i+50,x(a,i),y(a,i),a*3+i)
1160   next
1170   if mch(a)=1 then {
1180     sp_move(30+a,mx(a),my(a),6+a)
1190    } else {
1200     if mch(a)=2 then {
1210       sp_move(30+a,mx(a),my(a),6+a)
1220       sp_on(30+a)
1230       mch(a)=1
1240      }
1250    }
1260 endfunc
1270 /*
1280 /*                   ---===<<< chack() >
1290 func chack(x1,y1)
1300   if x1<10 or x1>235 or y1<10 or y1>224 then {
1310     return(-1)
1320    } else {
1330     return(0)
1340    }
1350 endfunc
1360 /*
1370 /*                   ---===<<< hit() >
1380 func hit(a,b)
1390 int x1,y1
1400   x1=mx(a)-x(b,0)
1410   y1=my(a)-y(b,0)
1420   if x1<5 and x1>-5 then {
1430     if y1<5 and y1>-5 then return(-1)
1440    }
1450 return(0)
1460 endfunc
1470 /*
1471 /*                   ---===<<< over() >
1480 func over(f)
1485   beep
1490   switch f
1500     case 99:print"Player 1(red) won!!":break
1510     case 98:print"Player 2(green) won!!":break
1520     case 197:print"Draw!!"
1530   endswitch
1540   repeat
1550   until strig(1)=1 and strig(2)=1
1553   cls:for a=0 to 1000:next
1555 endfunc
1560 /*
1570 /*                   ---===<<< sprite()
1580 func sprite()
1590 char sp(255)={
1600     1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,
1610     1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,
1620     1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,
1630     1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,
1640     1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,
1650     1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,
1660     1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0
1670    }
1680 char sp1(255)
1690 for a=0 to 5
1700   for b=0 to 255
1710     sp1(b)=sp1(b)+sp(b)
1720   next:print sp1(0)
1730   sp_def(a,sp1,1)
1740   if a<3 then {
1750     sp_color(a+1,rgb(31-a*7,0,0))
1760    } else {
1770     sp_color(a+1,rgb(0,31-(a-3)*7,0))
1780    }
1790 next
1800 char sp2(255)={
1810     0,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0,
1820     3,2,2,2,3,0,0,0,0,0,0,0,0,0,0,0,
1830     3,2,1,2,3,0,0,0,0,0,0,0,0,0,0,0,
1840     3,2,1,2,3,0,0,0,0,0,0,0,0,0,0,0,
1850     3,2,2,2,3,0,0,0,0,0,0,0,0,0,0,0,
1860     0,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0
1870    }
1880 sp_def(6,sp2,1)
1890 for a=0 to 255
1900   if sp2(a)<>0 then sp2(a)=sp2(a)+3
1910 next
1915 sp_def(7,sp2,1)
1917 print"Go!!"
1920 endfunc
1930 /*
1940 /*                   ---===<<< houkou2() >
2000 func houkou2(v,ho)
2010 int a
2020 a=ho-v
2030   if a<0 and a>-32 then v=v-1:return(v)
2040   if a<0 and a<-32 then v=v+1:return(v)
2050   if a>0 and a< 32 then v=v+1:return(v)
2060   if a>0 and a> 32 then v=v-1
2070 return(v)
2080 endfunc
```

プログラムに引数をつけよう！

そうそう。今月号のAX68K.Xには隠しオプションがあるのに気がつきましたか？　そう，リストを打ち込んでいたらわかったかもしれないけど，プログラムを実行する際にプログラム名のあとに数字を指定すると壁の数を変更できるのです。数字は壁の数になっていて，指定できるのは0～63個まで，何も指定しない(これを「デフォルトで」なんていいますね)と15枚になります。

たとえば，ほかのプログラムなどでも，

A>PRINT 読んでね.DOC

なんてプログラム名のあとにファイルを指定したりしますよね。そういうプログラムを作ったことはないですか？　なに，自分のプログラムでもやってみたいけど，どうやったら引数文字列にアクセスできるかわからなかった？

ま～かせてっ！　それではコマンドラインの引数文字列がどこに入っているのか，アセンブラ，C，それからX-BASICの場合でばっちり教えちゃいましょう。

アセンブラの場合

アセンブラで.xや.rになるプログラムを作ります。

そのプログラムを実行したときに，レジスタにいろいろな値が入っているのは知っていますか？　え，いつもプログラムの先頭からおもむろにメインを書いていたって。いけませんね～。そう，コマンド引数の文字列はプログラムが実行されたこのレジスタの中にちゃんと入っているのです。

もし，「C Compiler PRO-68K ver.2.0」を持っているなら，プログラマーズマニュアルの649ページを見てください。次のように書いてあるはずです。

●レジスタ
A0 = プログラムのメモリ管理ポインタのアドレス
A1 = プログラム(DATA，BSSを含む)の終わり+1のアドレス
A2 = プログラムに渡されたコマンドラインのアドレス
A3 = 環境のアドレス
A4 = プログラムの実行先頭アドレス
SR = モード(ユーザーモード)
USP = 親のスタックのまま
SSP = システムスタック

そう，A2が引数の文字列の入っている場所を指しているんですね。

実際にはA2の指しているアドレスからは図1のように最初の1バイトにその引数の大きさが何バイトかという情報が入っていて，2バイト目からはコマンドラインから切り出した引数が入っています。

たとえばTEST.Xというコマンドを作って，

A>TEST /? /N=Teatime

とコマンドラインから打ち込んだとしますね。するとTESTが実行されたとき，プログラムからは，図1のように見えるわけです。

だから，プログラムで引数の内容を知りたかったら，まず最初にA2の指しているアドレスの内容を保存しておく作業が必要になるんですね。また，あとでこの文字列を加工して使いたいというのであれば，たとえばSTRBUFという名前の領域を確保しておいて，

```
        addq.l  #1,a2
        lea     STRBUF,a3
loop:   move.b  (a2)+,(a3)+
        bne     loop
```

などととして，保存しておけばいいわけですね。そうそう，文字列の最後はちゃんとCで扱う文字列と同じように最後は#$00というデータが入るので，上のプログラムではそれを利用しています。

それにしても，ちゃんとコマンドラインの引数だけを切り出して渡してくれるのですから，Human68kも偉いですよね。

それから，

A>TEST /S=De | more

や，

A>TEST /S=De > test.log

のようにパイプ機能やリダイレクト機能を使っていても，プログラムからはちゃんとHuman68kは"/S=De"の部分だけを引数とみなしてそこだけ切り出してくれます。うーむ，偉いぞ，Human68k。

あ，パイプ機能やリダイレクト機能ってわかりますよね？　パイプ機能は2つのコマンドを|でつなげて最初のコマンドの標準出力の結果に次のコマンドを続けて実行してもらうことです。たとえば，

A>dir | more

で，dirで出力したディレクトリの内容を，画面に1ページずつ表示してくれます。

リダイレクションは標準入出力を変更します。たとえば，

A>dir > directory.log

とすれば，dirの結果を画面に出力せずに，directory.logというファイルに出力してくれるわけです。詳しくは「Human68kユーザーズマニュアル」を参照してくださいね。

でもってCの場合

Cってアセンブラに比べて配列なんかを使うのが格段に楽ですよね。それでってわけでもないんでしょうけど，引数の格納の仕方もばっちりと配列で用意してくれてます。しかもアセンブラではHuman68kが引数をコマンドから切り出してくれましたが，Cでは引数の要素を1つひとつ切り出してくれるのです。うーん，高級言語。

よく，Cのサンプルプログラムでメインルーチンの始まりに，

```
int main(int argc,char *argv [])
```

と書いてあるのを見たことがありませんか？　そう，Cではメインルーチンの関数の引数としてこのコマンドの引数が入ってきます。そこでargcには引数の個数，*argv[]には切り出した引数の要素が1つひとつ入るのです。ただ，ひとつ注意してほしいのは，Cの場合は切り出した引数のひとつ目はコマンドの名前が入ります。たとえば，

A>TEST /? /N=Teatime

argv[0]にはTESTが入っています。で，argv[1]には"/?"がargv [2] には"/N=Teatime"が入るわけです。で，引数の個数を表すargcには3が入ります。

先ほどのアセンブラの例のようにstrbufに引数をまとめて退避するには，

```
char strbuf[256]="";/*文字列退避領域*/
i=2;
if(argc >= 2)
    strcpy(strbuf,argv[1]);
while(i < argc) {
    strcat(strbuf," ");
    strcat(strbuf,argv[i]);
    i++;
}
```

とすれば大丈夫です。また，Cの引数の格納のされ方は，図2のようになります。

そういえば，引数はHuman68kの規則に従って“ ”などの文字で切り離されます。サンプルを使って，どう切り離されるのかいろいろ試してみてくださいね。

本当はないX-BASICの場合

X-BASICでは引数を調べる方法はありません。だって，X-BASICには引数っていう概念自体がないんだもん。

でも，これがコンパイルするとなると話は別です。X-BASICのプログラムをコンパイルするには，CCでCコンパイラを呼び出してましたよね。あれは，CコンパイラがX-BASICで書かれたリストをCのリストに直すBtoCコンパイラを呼び出して，できたCのリストをさらに.Xのファイルにコンパイルしているんですね。っていうことは話は簡単。そう，Cと同じなんです。

X-BASICのプログラムをコンパイルしてできたCのリストを見るとわかりますが，そのリストのメインルーチンでは，

```
int main(b_argc,b_argv)
int b_argc;
char *b_argv [];
```

と書かれているんです。だから，コンパイルするX-BASICのリストでもb_argcと*b_argv [] を参照すればいいわけです。

めでたしめでたし。

図1　引数の格納のされ方，アセンブラの場合

図2　Cで作ったプログラムの場合

Oh!X LIVE in '94

X68000・Z-MUSIC用
(SC-55対応)

「EUPHONY」より
PURE GREEN

Matsuo Naoki
松尾 直樹

X68000・Z-MUSIC ver.2.0用
(SC-55対応)

MUSICAL WORKS ©1993 NAMCO LIMITED「RIDGE RACER」より
Ridge racer(POWER REMIX)

Terada Koutarou
寺田 光太郎

ちょっと長めの2曲です。どちらもSC-55対応と，条件はややハードですが，まあ夏だし，許してね。持ってない人はくやし涙にくれていないで，ボーナスや夏休みのバイトでMIDI音源購入のご検討を。X68000ミュージシャンへの道を歩みましょう。

久々のカシオペア

今月の1曲目は，カシオペアのアルバム「EUPHONY」に収録されている「PURE GREEN」をお届けしましょう。スローテンポの落ち着いたリズムに優しげなギターの旋律が美しい，印象深い曲です。演奏にはZ-MUSICシステムとSC-55が必要となります。

この作品はキテます。だてにリストが長いのではありません。カシオペアのサウンドをここまで表現してしまった根性と，その完成度の高さには脱帽です。なに？ 長くて打ち込めん？ そんなことをいっちゃいけませんぜ。作るのにはもっと苦労したはずですからね。このヒューマニズム溢れる演奏に浸っていると，エディタを前に苦心を重ねている姿が目に浮かんでくるような気がします(そうだよね？)。

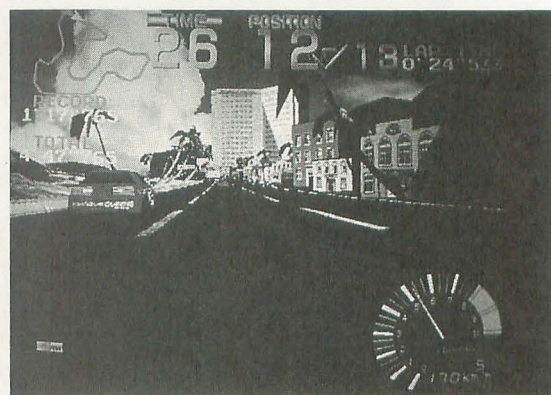
作者の松尾君は，いつも1000行を超えてしまうような曲を作っているそうで，この作品もリストを相当削った投稿用の特別バージョンだとか。「ほかの曲は大きすぎて掲載できないでしょう」という言葉に，自信と実力を感じます。初掲載でこのレベル，そのウラにはこんな秘密が隠されていたのですね。う～ん，このパワーを，夏バテ気味のボクらのカラダにも少々分け与えてほしい……。

それにしても，カシオペアのサウンドって幅がすごく広いんですよね。私もよくCDを聴きますが，ひとつのアルバムにもさまざまなサウンドが詰まっていて飽きさせません。時代にとらわれない反面，ときには流行をうまく噛み砕いて取り入れる斬新さ。私もたいへん気に入っているグループです。

EUPHONY

Are you ready?

次はナムコの業務用ドライビングゲーム「RIDGE RACER」より「Ridge racer (POWER REMIX)」です。こちらもSC-55対応です。

このゲームは昨年のOh!X12月号でも取材レポートという形でちらっと紹介されましたね。新型ハードウェア「SYSTEM22」で実現された「世界」には，いままでの常識を簡単に覆すほど強烈なものがありました。ドライビング感覚にまで突っ込むと，やはり「ゲームだな」という印象は拭い切れませんが，そもそも実車と比べてそういう意見を出したくなるってところで，すでに次元が違いますよね。

RIDGE RACER

作品の出来映えは，ファンの私も十分納得のいくレベルです。またあの日々を思い起こさせてくれますねぇ。いや～うれしい。シンセサイザ主体の曲を，うまくアレンジしています。この曲は，ブッ飛んだ「RIDGE RACER」のなかでは異色といえるほど「普通の音楽」ですから，安心して入力しましょう(どういう意味だ？)。

作者の寺田君は，なんとあの「X68000芸術祭」の音楽部門で全国大会に残ったというツワモノ。これまた期待の大型新人登場です。うれしいことに，3月号のリクエストに応えてこの曲を投稿してくれたようですね。感謝感謝。じゃあ，またリクエストをしてみようかな。同じカーレースからの選曲で「アウトランナーズ」なんかどうでしょうか(こらこら)。

ところで，このゲームのフルスケール版で練習に練習を重ねると，教習所へ通わなくても免許が取れるようになるかもしれないってホント？ 教習所とどっちがお金かかるかなぁ……。 (T.F)

協力 株式会社ナムコ

リスト1 PURE GREEN

```
  1: .COMMENT -- PURE GREEN --  CASIOPEA  for SC-55   by  ﾏｯﾁｭﾝ
  2: (1)(B1)
  3: (M1,5000)(AMIDI1,1)
  4: (M2,9000)(AMIDI2,2)
  5: (M3,9830)(AMIDI3,3)
  6: (M4,6000)(AMIDI4,4)
  7: (M5,5000)(AMIDI5,5)
  8: (M6,6000)(AMIDI6,6)
  9: (M7,3000)(AMIDI7,7)
 10: (M8,4800)(AMIDI8,8)
 11: (M9,3000)(AMIDI9,9)
 12: (M10,5000)(AMIDI10,10)
 13: (M11,1000)(AMIDI11,11)
 14: (M12,2000)(AMIDI12,12)
 15:
 16: /EDなんかの二重化を使えば、入力が楽でしょう。
 17: .ROLAND_EXCLUSIVE $10,$42={$40,$00,$7F,$00}  /音源初期化
 18: .ROLAND_EXCLUSIVE $10,$42={$40,$11,$30,$38}  /VibratoRate
 19: .ROLAND_EXCLUSIVE $10,$42={$40,$19,$30,$38}  /VibratoRate
 20: .ROLAND_EXCLUSIVE $10,$42={$40,$11,$31,$4F}  /VibratoDepth
 21: .ROLAND_EXCLUSIVE $10,$42={$40,$19,$31,$4F}  /VibratoDepth
 22: .ROLAND_EXCLUSIVE $10,$42={$40,$11,$37,$52}  /VibratoDelay
 23: .ROLAND_EXCLUSIVE $10,$42={$40,$19,$37,$52}  /VibratoDelay
 24: .ROLAND_EXCLUSIVE $10,$42={$40,$11,$34,$3F}  /attack
 25: .ROLAND_EXCLUSIVE $10,$42={$40,$19,$34,$3F}  /attack
 26: .ROLAND_EXCLUSIVE $10,$42={$40,$11,$35,$42}  /Decay
 27: .ROLAND_EXCLUSIVE $10,$42={$40,$13,$35,$43}  /Decay
 28: .ROLAND_EXCLUSIVE $10,$42={$40,$19,$35,$50}  /Decay
 29: .ROLAND_EXCLUSIVE $10,$42={$40,$1B,$35,$48}  /Decay
 30: .ROLAND_EXCLUSIVE $10,$42={$40,$11,$32,$50}  /cutoff freq
 31: .ROLAND_EXCLUSIVE $10,$42={$40,$12,$32,$3A}  /cutoff freq
 32: .ROLAND_EXCLUSIVE $10,$42={$40,$13,$32,$44}  /cutoff freq
 33: .ROLAND_EXCLUSIVE $10,$42={$40,$15,$32,$3D}  /cutoff freq
 34: .ROLAND_EXCLUSIVE $10,$42={$40,$19,$32,$50}  /cutoff freq
 35: .ROLAND_EXCLUSIVE $10,$42={$40,$1B,$32,$44}  /cutoff freq
 36: .ROLAND_EXCLUSIVE $10,$42={$40,$11,$33,$45}  /resonance
 37: .ROLAND_EXCLUSIVE $10,$42={$40,$12,$33,$3A}  /resonance
 38: .ROLAND_EXCLUSIVE $10,$42={$40,$15,$33,$3B}  /resonance
 39: .ROLAND_EXCLUSIVE $10,$42={$40,$19,$33,$4C}  /resonance
 40: .ROLAND_EXCLUSIVE $10,$42={$40,$1B,$33,$46}  /resonance
 41: .ROLAND_EXCLUSIVE $10,$42={$40,$11,$36,$48}  /release
 42: .ROLAND_EXCLUSIVE $10,$42={$40,$13,$36,$3E}  /release
 43: .ROLAND_EXCLUSIVE $10,$42={$40,$19,$36,$44}  /release
 44: .ROLAND_EXCLUSIVE $10,$42={$40,$1B,$36,$44}  /release
 45: .ROLAND_EXCLUSIVE $10,$42={$40,$1A,$15,$02}  /Use Drumpart
 46:
 47: .sc55_v_reserve={2,3,6,3,1, 1,2,1,2,2, 1,0,0,0,0, 0}
 48: .sc55_print"PURE GREEN"
 49: .SC55_reverb $10={4,4,2,118,88,12,00}
 50: .SC55_chorus $10={2,0,66,5,4,4,15,00}
 51:
 52: (t1)t104o5@v108q7@p56@g12@u98116@27k-12"1     / Guitar
 53: (t2)o3q5@v118@p57@g12@u97116i127,0@21             / Clavinet
 54: (t3)o4@q3@v74@p60@g12@u80116@53           / Chorus
 55: (t4)o5@q2@v96@p64@g12@u110116@49          / Strings
 56: (t5)o3q7@v88@p70@g12@u95116@13            / Marimba
 57: (t6)o2q7@v115@p64@g12@u110116@38          / Bass
 58: (t7)o4q7@v127@p69@g12@u110116@70          / Horn
 59: (t8)o4q7@v116@p64@g12@u110116@19k-12@c11       / Low Strings
 60: (t9)o5@v73Q7@p68@g12@U98116@25"1k-12@m20       / Guitar
 61: (t10)o3@v105@r1@u100116@1                 / drum
 62: (t11)o2@v110@r1@u127116@9               / drum(kick)
 63: (t12)o4q7@v95@g12@u95116@124 x$40,$1B,$1C,00 / Birds
 64:                                /↑ﾗﾝﾀﾞﾑ  ﾊﾟﾝ
 65: (t1)@i$41,$10,$42@e90,57        y$7e,1
 66: (t2)@i$41,$10,$42@e68,38
 67: (t3)@i$41,$10,$42@e80,38
 68: (t4)@i$41,$10,$42@e90,30
 69: (t5)@i$41,$10,$42@e50,10
 70: (t6)@i$41,$10,$42@e20,36                y$7e,1
 71: (t7)@i$41,$10,$42@e108,25w
 72: (t8)@i$41,$10,$42@e65,80
 73: (t9)@i$41,$10,$42@e84,50        y$7e,1
 74: (t10)@i$41,$10,$42@e88,40       @y$1d,37,122
 75: (t11)@i$41,$10,$42@e36,50
 76: (t12)@i$41,$10,$42@e116,10w
 77:
 78: /以降(t11)で、Wコマンドの制御をしますので要確認!
 79: /------------------------  A  ------------------------------
 80:
 81: (t1,9)r1  r1r1r1r1   r1r1r1r1
 82:
 83: (t2)r1  z100,95,88,95|:4r8'b8<dg'r'b<dg'r8'b<dg'r8'b<dg'r4
 84: r8'a8<f+a'r'a<f+a'r8'a<f+a'r8'a<f+a'r4:|
 85:
 86: (t3)r1  |:4'd1gb'  'd1f+a':|
 87:
 88: (t4)r1  r1r1  z100,115d1  <d1>>
 89: z85,95,105,110,105,115,120,125,98,115,110,104r8d8e8g8<d4.e8
 90: f+2g4a4  >b2<a4.g8  f+2>z95,105,98'a8.<df+''a8.<dg''a8<df+'
 91:
 92: (t5)r1  z78,88,80,86|:4|:4dbeg:|  |:4daef+:|:|
 93:
 94: (t6)r1  z110,95,110,110d8r8.ddrd8r4.  z110,95,110,115,120
 95: d8r8.ddra4<d4>  z110,95,110,110d8r8.ddrd8r4.
 96: z110,95,110,115,120d8r8.ddr>a4d4<  z110,95,110,110
 97: d8r8.ddrd8r4.  z110,95,110,115,120  d8r8.ddra4<d4>
 98: z110,95,110,110d8r8.ddrd8r4.>  z110,95,110,110,115,120
 99: d8r8.<dar>d8.a8.<d8
100:
101: (t8)r*960 z85,95,105,110,105,115,120,125,100,115,110,104
102: r8d8e8g8<d4.e8  f+2g4a4  >b2<a4.g8  f+2r2
103:
104: (t10)r2^8.z120,119c>g4<  z100,,115,100,100,110
105: g4>'c+f+'f+f+8f+4'c+4f+' |:6f+4'c+f+'f+f+8f+4'c+4f+':|
106: f+4'c+f+'f+f+8a+4'c+4f+'
107:
108: (t11)r1  |:8c4.c8c2:|
109:
110: /------------------------  B  ------------------------------
111:
112: (t1,9)|:2o5z110,100,110r4g8d8g8.@u115a&a8&b8
113: z101,114,108,102,98,,104,112a2a8.g8.f+8  e2&e8.d8.e8
114: @u100e*2&f+*190  z110,100,110r4g8d8g8.@u115a&a8&b8
115: @u100b*2&a*94z110,105,100,96,,88a8.g8.f+8
116: e2&e8.d8.@u108f*2&f+*22
117: |1z114,100a2.a4  @u88<c+*2&d*190>  r1r1r1:|
118: |2@u100a2.r4
119:
120: (t2)|:2o4z100,95,88r8'c8eg'r'ceg'r8'ceg'r4..>
121: r8'b8<df+'r'b<df+'r8'b<df+'r4..
122: r8'b-8<deg'r'b-<deg'r8'b-<deg'r4..
123: r8'a8<df+'r'a<df+'r8'a<df+'r4..:|
124: |:2o3z100,95,88,95r8'b8<dg'r'b<dg'r8'b<dg'r8'b<dg'r4
125: r8'a8<df+'r'a<df+'r8'a<df+'r8'a<df+'r4:|
126: |:2o4z100,95,88r8'c8eg'r'ceg'r8'ceg'r4..>
127: r8'b8<df+'r'b<df+'r8'b<df+'r4..
128: r8'b-8<deg'r'b-<deg'r8'b-<deg'r4..
129: |1r8'a8<df+'r'a<df+'r8'a<df+'r4..:||2r1
130:
131: (t3)|:2o4'c1eg'  >'b1<df+'  'b-1<de'  'a1<df+':|
132: |:2o4'd1gb'  'd1f+a':|  o4'c1eg'  >'b1<df+'  'b-1<de'
133: 'a1<df+'  <'c1eg'  >'b1<df+'  'b-1<de'  'a2<df+'r2
134:
135: (t4)@u80|:2'b1<de'  'd1f+a''e1gb-''d1f+a':|
136: z110,114,118,120,122,125,127,120,115,110,120,113
137: b2.<c+4  d2e4f+4  g2a4.g8  f+2>'a8.<df+''a8.<dg''a8<df+'
138: @u86'b1<de'  @u80'd1f+a''e1gb-''d1f+a'  @u65'b1<de'
139: @u80'd1f+a''e1gb-'
140: 'd2f+a'z100,115,104,106,110,115,120,122de8f+{a<def+a}4
141:
142: (t5)|:2o3|:4d<c>gb:||:4dbf+a:||:4db-eg:||:4daef+:|:|
143: |:2|:4dbeg:||:4daef+:|:||:2o3|:4d<c>gb:||:4dbf+a:|
144: |:4db-eg:||:4daef+:|:|
145:
146: (t6)|:2o2z110,95,110,110c8r8.ccrc8r4.  >b8r8.bbrb8r4.
147: b-8r8.b-b-rb-8r4.  a8r8.aara8r4.:|  o2z110,95,110,110
148: d8r8.ddrd8r4.  z110,95,110,115,120d8r8.ddra4<d4>
149: z110,95,110,110d8r8.ddrd8r4.  z110,95,110,100,110,120
150: >d8r8.<dar>d8.a8.<d8  |:2o2z110,95,110,110c8r8.ccrc8r4.
151: >b8r8.bbrb8r4.  b-8r8.b-b-rb-8r4.  a8r8.aara8r4.:|
152:
153: (t8)o4r*1536   z110,114,118,120,122,125,127,120,115
154: b2.<c+4  d2e4f+4  g2a4.g8  f+2@u117r2  o5r1r1r1
155: @u122r2@a0,10,20,30,40,50,60,70'a4<df+'&
156: @a75,80,85,90,95,100,105,110'a8.<df+'r@a127  r1r1r1r1
157:
158: (t10)z100,,115,100,100,110|:2o3a4>'c+f+'a+f+8f+4'c+4f+'
159: |:6f+4'c+f+'a+f+8f+4'c+4f+':| z105,98,115,98,,80,,98,115,80
160: |1f+4'c+f+'f+a+8f+8f+f+'c+f+'f+a+8  z100,,115,100,100,110
161: |:3f+4'c+f+'a+f+8f+4'c+4f+':|  z100,100,115,100,80
162: f+4'c+f+'a+f+8f+4@u120<d8c8>:|
163: |2z100,100,115,100,80f+4'c+f+'a+f+8f+4@u120<c8>g8
164:
165: (t11)|:11c4.c8c2:|  c4.c8r2  |:8c4.c8c2:|
166:
167: /------------------------  C  ------------------------------
168:
169: (t1,9)o5"1r4z115,101,105,110,105,,98b8f+8b8.<c+8.c+*2&d*22
170: d2z114,,111,109,105,100,110,114,116,116c+*2&d*34c+8.>b8
171: a4.e8a8.b8.b*2&<c*22>
172: @u110b*2&<c*94&c8.@u106c*2&>b*34z102,96a8
173: g2z98,105,108,110,107,114,+2,118,120,122,98,110,118g8.a8.b8
174: b2b4<c+4  d8.d8.e8e2  r4>a4<e4f+4
175:
176: (t2)o4@u88|:8'f+8g+b<d':|  |:8'e+8gb<d':|  |:8'e8ga<d':|
177: |:8'd+8f+a<c':|  |:8'e8gb<d':|  |:8'd8gb':|  |:8'e8f+a<d':|
178: 'e8f+a<d''e8f+a<d''e8f+a<d''e8f+a<d'z98,108'e4a<c''f+4a<d'
179:
180: (t3)o3@u60'd1f+g+b<c+d'  'c+1egb<c+d'  'c1egb<cd'
181: >'b1<d+f+ab<c'  <'d1egab<d'  'd1egb<c+d'o4  z100,95,100,98
182: 'e8.f+a<d''e8.f+a<d''e8f+a<e''e2f+a<e'  r1
183:
184: (t4)o6z114,117,114,108,110,112,115,110d1  d4c+4>g4g+4
185: a4.b8a2  z114,110,105,108,110,106,109,112,104,108,110
186: <c4>b4f4f+4  g2g4a4  b4d4g4b4  z112,105,110,115<d8.d8.e8e2>
187: r4z105,110,107a4<e4f+4
188:
189: (t5)o3@u80'd1f+g+b<c+d',12  'c+1egb<c+d'  'c1egb<cd'
190: >'b1<d+f+ab<c'  <'d1egab<d'  'd1egb<c+d'o4  z100,95,100,98
191: 'e8.f+a<d',0'e8.f+a<d''e8f+a<e''e2f+a<e'  r1
192:
193: (t6)o1@u110g+2.r8g+8  <c+2.r8c+8  c2.r8c8  >b2.r8b8
194:  e2.r8e8  a2.r8a8  <f+2.r8f+8  >z110,115,120f+2a4<d4
195:
196: (t8)o6@u108r1r1r1r1r1r1  a1  r1
197:
198: (t10)z110,95,,,110,95,,,,110,95
199: o3b>f+f+f+a+8f+8f+f+f+f+a+8f+8
200: |:6f+f+f+f+a+8f+8f+f+f+f+a+8f+8:|o3
201: z114,95,,,108,95,,110,110,98,,110g>f+f+f+a+8f+f+<g8d8>dd<c8
202:
203: (t11)r1r1r1  r1r1r1r1  c2c2
204:
205: /------------------------  D  ------------------------------
206:
207: (t1,9)o6z125,114,118,114,-4,114,,122,110,+5,120,,108,+2,115
208: d4.d8d8.c+8.>b8  <c+*2&d*94e8.d8.e8
209: e*2&f+*70>a8a8.b8.@u120b*2&<c+*22
210: z108,113,115,118,122,114,116,112,106c+2>b8.<c+8.f+8
211: a4.a8a8.g8.f8  z114,,100,118,114,106,110,102,102
212: g*2&a*70>a8<g8f+8e8f+d&  d1>z100,118,114,106,110,102,102
213: r4.a8<a8f+8e8f+d&  d1  r1
214:
```

```
215: (t2)r1r1r1r1r1  r1r1r1r1r1
216:
217: (t3)@u75o4'f+1ab<d'  'e1ab-<d'  'f+1a<d'  'd+1f+a<c+'
218: 'a1b-<df'  'g1b<d'  'd1gb' 'df+1a'  'd1gb'  'df+1a'
219:
220: (t4)o4@u88'f+1ab<d' 'e1ab-<d'  'f+1a<d'  z90,114,118,120,88
221: 'd+2f+a<c+''b8.>b'<c+8.f+8>. 'a1b-<df'  @u88'g1b<d'
222: z105,108,113,116,120,122,124,104,115,110,108,110,115,105
223: r8d8g8b8<d4.e8  f+2a2  >b2<a4.g8  f+2'd8.f+''d8.g''d8f+'
224:
225: (t5)o3z78,88,80,86  |:4dbeg:||:4db-eg:||:4f+<d>ab:|
226: |:4f+<c+>ab:|  |:4f<d>gb-:|  |:4g<d>ab:|
227: |:2|:4dbeg:||:4daef+:|:|
228:
229: (t6)o1z115,110,95,110g8r8.ggrg8r4.  <c8r8.ccrc8r4.>
230: f+8r8.f+f+rf+8r4.  z115,-5,95,110,122,+3b8r8.bbrb8.<f+8.b8>
231: z115,110,95,110b-8r8.b-b-rb-8r4.
232: @u110a8r8<@u118@b-3415,0,1a*16&@b0a*128  o2z115,110,95,110
233: d8r8.ddrd8r4.  z115,110,95,118,122d8r8.ddra4<d4>
234: z115,110,95,110d8r8.ddrd8r4.  >z115,110,95,118,120,127
235: d8r8.<d>drd8.a8.<d8
236:
237: (t8)o4r1r1r1  z114,118,120,124r2b8.<c+8.f+8  a1  o4r1
238: z105,108,113,116,120,122,124,104,115,110,108r8d8g8b8<d4.e8
239: f+2a2  >b2<a4.g8  f+2r2
240:
241: (t10)o3z110,110,115,100,100,110a4>'df+'a+f+8f+4'df+4'
242: z110,110,108,110,110f+8.<c>'df4'f+4'df+4'
243: z110,110,115,100,100,110f+4'df+'a+f+8f+4'df+4'
244: z110,120,110,118,110,110,115,108f+8.<c>d<c>f8<a8.b>d8<g8
245: z108,95,110,115,110,110,95,,110,118,100
246: a8>f+8'df+'f+a+8f+8f+f+d<d>a8   z95,110d*2d*46<a2>g4
247: z110,110,115,110,105<b4>'c+f+'f+f+8f+4'c+4f+'
248: |:2f+4'c+f+'f+f+8f+4'c+4f+':|
249: z110,105,115,110f+4'c+f+'a+f+8@u108<a8.g>d8<g8
250:
251: (t11)c4rcc8c2  c4.c8c2  c4rcc8c2  c4.c8c8.c8.c8
252: c4rcc8c4..c  r4c2.  c4.c8c2  c4.c8c2  c4.c8c2 c4.c8c8.c8.c8
253:
254: /---------------------  E  ------------------------
255:
256: (t1,9)o5z110,100,110r4g8d8g8.@u112a&a8@u118a*2&b*22
257: z101,114,108,102,98,,104,112a2a8.g8.f+8  e2&e8.d8.e8
258: @u100e*2&f+*190  z110,100,110,112r4g8d8g8.a8.@u118a*2&b*22
259: @u100b*2&a*94z110,105,100,106,100,107a8.g8.f+8
260: e2e8.f+8.@u110g8&  g1
261:
262: (t2)o4z100,95,88r8'c8eg'r'ceg'r8'ceg'r4..>
263: r8'b8<df+'r'b<df+'r8'b<df+'r4..
264: r8'b-8<deg'r'b-<deg'r8'b-<deg'r4..
265: r8'a8<df+'r'a<df+'r8'a<df+'r4.. <r8'c8eg'r'ceg'r8'ceg'r4..>
266: r8'b8<df+'r'b<df+'r8'b<df+'r4..
267: r8'b-8<deg'r'b-<deg'r8'b-<deg'r4..  r1w
268:
269: (t3)@u88o4'c1eg'  >'b1<df+'` 'b-1<de'  'a1<df+'
270: <'c1eg'  >'b1<df+'  'b-1<de'r1w
271:
272: (t4)o4@u80'b1<de'  'd1f+a''e1gb-''d1f+a'  @u65'b1<de'
273: @u80'd1f+a''e1gb-'r1
274:
275: (t5)o3|:4d<c>gb:||:4dbf+a:||:4db-eg:||:4daef+:||:4d<c>gb:|
276: |:4dbf+a:|  db-eg db-eg db-eg db-@u88'g8b-<d'&  'g1b-<d'
277:
278: (t6)o2z110,95,110,110c8r8.ccrc8r4.  >b8r8.bbrb8r4.
279: b-8r8.b-b-rb-8r4.  a8r8.aara8r4.  <c8r8.ccrc8r4.
280: >b8r8.bbrb8r4.  b-8r8.b-b-rb-8r4@u114a8&  a1
281:
282: (t7)z88,86,80,82,84,86,94,85,80,75,78,82,86,90,94
283: r4f+*3e*3d*6ef+gagab-<cdef+
284:
285: (t8)o5r1r1r1  @u122r2@a0,10,20,30,40,50,60,70'a4<df+'&
286: @a75,80,85,90,95,100,105,110'a8.<df+'r@a127  r1r1r1r1
287:
288: (t10)z100,,115,100,100,110o3a4>'c+f+'f+f+8f+4'c+4f+'
289: |:5f+4'c+f+'a+f+8f+4'c+4f+':|  z110,110,115,110,100,114
290: f+4'c+f+'a+f+8f+4'c+8f+'<g8  z100,115,114,110r2d*2d*46c8>g8
291:
292: (t11)|:6c4.c8c2:|  c4.c8c4.c8  w7r1
293:
294: /---------------------  F  ------------------------
295:
296: (t1)o4z35,45,55,65,65ab<dg&g2>r4  ga<df+&f+2>r4 ab<dg&g2>r4
297: r1  o4|:2ab<dg&g2>r4  ga<df+&f+2>r4:|
298: o5|:4ab<dg&g2>r4  ga<df+&f+2>r4:|
299:
300: (t9)r1w
301:
302: (t4)r*1344 z98,105,110,115,,78r1 |:4o3ab<dg&g2r4  'd1f+a':|
303:
304: (t5)o3z78,88,80,86|:8|:4dbeg:||:4daef+:|:|
305:
306: (t6)o2z110,95,110,110|:3d8r8.ddrd8r4.:| z110,95,110,116,122
307: d8r8.ddra4<d4>  o2z110,95,110,110|:3d8r8.ddrd8r4.:|
308: z110,95,110,120,94,,88d8r8.dd8>a4(g8a)<c8  z110,95,110,110
309: |:3d8r8.ddrd8r4.:|  z110,95,110,100,95,112d8r8.ddr>b8.a8.b8
310: <z110,95,110,110|:3d8r8.ddrd8r4.:|  z110,95,110,96,,104,110
311: d8r8.ddr@b-3145,0,1b&@b0bb8@b0,-8191,24a4@b0
312:
313: (t7)z100,92,84,78,70g4.(f+d>b)8g2
314: z88,72,76,80,84,88,90,94,108,92,94
315: r8.d.e*6f+*6a*6<d*6e*6f+*6a*6<d4.c+d  z110,100,96c+4.>b8b2
316: z115,108,104,108,109,111,115,118<r4(rc+>f+f+gggab)2.
317: z112,82,84,86,88,82,86,88,78,74,68,,81,74,74
318: a4.ef+gaef+ged>a&  a8<d>a&a4r2
319: z92,76,78,80,82,84,86,88,90,92,90,92,94,96
320: r8f+d+ef+gab<d+f+af+gab-
321: z110,100,90,80,70,88,118,116,110,,114,116
322: b8af+edr<d*2e*10dc+rc+&(c+de)4>  z110,100,94,90,94
323: a8ba2.<e*6f+*6  z114,90,98,92e2.r>a<d>b
324: z112,88,92,96,114,114g4.(ef+g)8<d4&d>z83,88,93,98c+de
325: f+ef+gag+aa+ba+b<c+dc+de  z98,118
326: e*2f+*82z92,82,72,62e*6d*6c+>ar4z72,82,88(ef+a)8
327: z109,81,84,104,80,84,88,110,86
328: <d4>d*2e*2f+*44e*2f+*2g*2a*78f+  z118,82,84,91,81,95
329: <d4.c+ec+8.>b<c+4  z78,84,82,78,76,80,84,94,104
330: r>a<d>gf+ff+ga4<d4  <@u48d1
331:
332: (t8)o6r*2112  r2.¥-25r4  @u127d1&  d1&  d1&  d1
333:
334: (t10)z104,100,115,110,110,100o3a4>'c+f+'f+a+8f+4'c+4f+'
335: z110,100,115,110,110,100|:2f+4'c+f+'a+f+8f+4'c+4f+':|
336: z110,100,115,110,110,100,100f+4'c+f+'a+f+8f+8.f+'c+4f+'
337: z110,100,115,110,110,100|:3f+4'c+f+'a+f+8f+4'c+4f+':|
338: z110,100,115,110,,95,115,100,95,110
339: f+4'c+f+'a+f+8f+8f+f+'c+f+'f+a+8  z110,100,115,110,110,100
340: |:5f+4'c+f+'a+f+8f+4'c+4f+':|  z110,100,115,110,,100,100
341: f+4'c+f+'a+f+8f+8.f+'c+4f+'  z110,100,115,110,,100
342: f+4'c+f+'a+f+8f+4'c+4f+'  z110,100,115,110,110,122,114,104
343: f+4'c+f+'a+f+8f+8.<dc8>g8
344:
345: (t11)w12|:16c4.c8c2:|w9
346:
347: (t12)@u50g4d4r2  r4f+4g2  r4f+2c4  @u55r2b2  r1  r2e2
348: @u62r4g2r4  r2f+2 r4g4r2  r2a2  r1  g2r2  r2d2
349:
350: /------------------------------------------------
351: /-------------------  Repeat  --------------------
352: /------------------------------------------------
353:
354: /---------------------  C  ------------------------
355:
356: (t1,9)o5r4z115,101,105,110,105,,98b8f+8b8.<c+8.c+*2&d*22
357: d2z114,,111,109,105,100,110,114,116,116c+*2&d*34c+8.>b8
358: a4.e8a8.b8.b*2&<c*22>
359: @u110b*2&<c*94&c8.@u106c*2&>b*34z102,96a8
360: g2z100,105,+3,110,-3,114,116,118,+2,122,104,110,118g8.a8.b8
361: b2b4<c+4  d8.d8.e8e2  r4>a4<e4f+4
362:
363: (t2)o4@u88|:8'f+8g+b<d':|  |:8'e+8gb<d':|  |:8'e8ga<d':|
364: |:8'd+8f+a<c':|  |:8'e8gb<d':|  |:8'd8gb':|  |:8'e8f+a<d':|
365: 'e8f+a<d''e8f+a<d''e8f+a<d''e8f+a<d'z98,108'e4a<c''f+4a<d'
366:
367: (t3)o3@u60'd1f+g+b<c+d'  'c+1egb<c+d'  'c1egb<cd'
368: >'b1<d+f+ab<c'  <'d1egab<d'  'd1egb<c+d'o4  z100,95,100,98
369: 'e8.f+a<d''e8.f+a<d''e8f+a<e''e2f+a<e'  r1
370:
371: (t4)o6z114,117,114,108,110,112,115,110d1  d4c+4>g4g+4
372: a4.b8a2  z114,110,105,108,110,106,109,112,104,108,110
373: <c4>b4f4f+4  g2g4a4  b4d4g4b4  z102,95,100,105
374: 'a8.<d'<d8.e8e2>  r4z95,100,97a4<e4f+4
375:
376: (t5)o3@u80'd1f+g+b<c+d',6  'c+1egb<c+d'  'c1egb<cd'
377: >'b1<d+f+ab<c'  <'d1egab<d'  'd1egb<c+d'o4  z100,95,100,98
378: 'e8.f+a<d',0'e8.f+a<d''e8f+a<e''e2f+a<e'  r1
379:
380: (t6)o1@u110g+2.r8g+8  <c+2.r8c+8  c2.r8c8
381: >b2.r8b8  e2.r8e8  a2.r8a8 <f+2.r8f+8  >z110,+5,120f+2a4<d4
382:
383: /ﾋﾄﾂ F ﾉ ﾘﾘｰｽｵﾝ ﾄｼﾃ ｶｸﾎｼﾀﾀﾒ ｽﾞﾚﾃﾏｽ
384: (t8)o6@u108r1r1  r1r1r1a1  r1
385:
386: (t10)z112,95,,,110,95,,,,110,95
387: o3b>f+f+f+a+8f+8f+f+f+a+8f+8
388: z100,95,,,110,95,,,,110,95|:6f+f+f+f+a+8f+8f+f+f+a+8f+8:|
389: z110,95,,,108,95,,110,,100,,+9o3g>f+f+f+a+8f+f+<g8d8>dd<c8
390:
391: (t11)w2w3r1r1r1  r1r1r1r1  c2c2
392:
393: /-------------  D - toCODA + CODA  ------------------
394:
395: (t1,9)o6z125,114,+4,114,-4,114,,+8,110,+5,120,,108,110,115
396: d4.d8d8.c+8.>b8  <c+*2&d*94e8.d8.e8
397: e*2&f+*70>a8a8.b8.@u120b*2&<c+*22
398: z108,113,115,118,122,114,116,112,106c+2>b8.<c+8.f+8
399: a4.a8a8.g8.f8  z114,,100,118,114,106,110,102,102
400: g*2&a*70>a8<g8f+8e8f+d&  d2.@u112q6e*2&f+*22&e*6&f+*18&
401: f+4.q7z95,98,101,98,104,106,110,114,114ef+gef+ga8a*2&b*22
402:
403: (t2)r1r1r1r1r1  r1r1r1
404:
405: (t3)@u75o4'f+1ab<d'  'e1ab-<d'  'f+1a<d'  'd+1f+a<c+'
406: 'a1b-<df'  'g1b<d'  'd1gb'  'd4f+a'r2.
407:
408: (t4)o4@u88'f+1ab<d'  'e1ab-<d'  'f+1a<d' z90,114,118,120,88
409: 'd+2f+a<c+'b8.<c+8.f+8>  'a1b-<df'  @u88'g1b<d'
410: z100,105,+3,113,116,120,+2,124,114,100,105r8d8g8b8<d8e8g8b8
411: <d4>>a4'g4<ce''a4<df+'
412:
413: (t5)o3z78,88,80,86  |:4dbeg:||:4db-eg:||:4f+<d>ab:|
414: |:4f+<c+>ab:||:4f<d>gb-:||:4g<d>ab:||:4dbeg:||:4daef+:|
415:
416: (t6)o1z115,110,95,110g8r8.ggrg8r4.  <c8r8.ccrc8r4.>
417: f+8r8.f+f+rf+8r4.  z115,-5,95,110,122,+3b8r8.bbrb8.<f+8.b8>
418: z115,110,95,110b-8r8.b-b-rb-8r4.
419: @u110a8r8<@u118@b-3415,0,1a*16&@b0a*128  z115,110,95,110
420: o2d8r8.ddrd8r4.  >z115,110,95,118,122d8r8.ddra4<d4>
421:
422: (t8)o4r1r1r1  z114,118,120,124r2b8.<c+8.f+8  a1  r1
423: z100,+5,108,113,+3,120,+2,124,114,100,+5r8o4d8g8b8<d8e8g8b8
424: <d4>>a4<c4f+4
425:
426: (t10)o3z110,110,115,100,100,110a4>'df+'a+f+8f+4'df+4'
427: z110,,108,110,110f+8.<c>'df4'f+4'df+4'  z110,,115,100,,110
428: f+4'df+'a+f+8f+4'df+4'  z110,120,110,118,110,110,115,108
429: f+8.<c>d<c>f8<a8.b>d8<g8 z108,95,110,+5,110,,95,,110,+8,100
430: a8>f+8'df+'f+a+8f+8f+f+d<d>a8  z95,110d*2d*46<a2>g4
431: z105,110,115,100,110,100<a4>'df+'a+f+8f+4'df+4'
432: z110,95,115,110,115,110,,104,+4,110f+4'df+'a+f+8ddr8d<dd>a+
433:
434: (t11)c4rcc8c2  c4.c8c2  c4rcc8c2  c4.c8c8.c8.c8 c4rcc8c4..c
```

```
435: r4c2.  c2c2  c4.c8c2
436:
437: /------------------------  G  -----------------------------
438:
439: (t1,9)z118,,88,115,110a2&a8>a8<a8g+8
440: z114,98,114,110,95,120,98g8gf+8.d8>a+4<d8e@u110e*2&f+*10&
441: f+4z88,104,104r>ara&a2  z78,80,82,84,86,88,90,92,94,96,98
442: r4ref+gab<cdef+ga@u108a*2&b*94&b8@u104a*2&b*34@u95e*2&f+*34
443: z102,88,104,106,110,102,118,118e2e8.f+8.g8  a8f+a&a2.
444: >z95,115,125r4a4<e4f+4
445: z122,118,120,,116,110,+6,,120,114,118d4.d8c+*2&d*34c+8.>b8<
446: c+*2&d*94e8.d8.e8  z126,,102,108,112,106,,100,,96,106,110
447: e*2&f+*70>a8a8.b8.b*2&<c+*22>  b*2&<c+*94>b8.<c+8.f+8
448: z114,104,112,104,100a4.a8a8.g8.f8
449: z108,,92,124,120,118,110,102,102g*2&a*70>a8<g8f+8e8f+d&
450: d1>  z92,124,120,118,110,102,102r4.a8<a8f+8e8f+d&  d1  r1
451:
452: (t2)|:2o4z100,95,88,95r8'g8b<d'r'gb<d'r8'gb<d'r8'gb<d'r4
453: r8'e8ab-<d'r'eab-<d'r8'eab-<d'r8'eab-<d'r4
454: r8'f+8a<d'r'f+a<d'r8'f+a<d'r8'f+a<d'r4  z100,95,100,110
455: r8'f+8a<d'r'f+a<d'r8'e4a<c''f+4a<d':|  z100,95,88,95
456: r8o4'g8b<d'r'gb<d'r8'gb<d'r8'gb<d'r4
457: r8'e8ab-<d'r'eab-<d'r8'eab-<d'r8'eab-<d'r4
458: r8'f+8a<d'r'f+a<d'r8'f+a<d'r8'f+a<d'r4
459: r8'd+8f+a<c+'r'd+f+a<c+'r8'd+f+a<c+'r8'd+f+a<c+'r4
460: r8'f8ab-<d'r'fab-<d'r8'fab-<d'r8'fab-<d'r4
461: r8'g8b<d'r'gb<d'r8'gb<d'r8'gb<d'r4
462: r8'd8gb'r'dgb'r8'dgb'r8'dgb'r4
463: r8'd8f+a'r'df+a'r8'df+a'r8'df+a'r4
464:
465: (t3)o4@49~18|:2@u88'g1b<d'  'e1ab-<d'  'f+1a<d' z100,95,110
466: 'f+2a<d''e4a<c''f+4a<d':|  @u88'g1b<d'  'e1ab-<d'  'f+1a<d'
467: 'd+1f+a<c+'  'f1ab-<d'  'g1b<d'  'd1gb'  'd1f+a'z100,108,98
468: >'b-2<dfb-'<'c2eg<c'  ¥35'd1f+a<d'&  'd1f+a<d'
469:
470: (t4)z110,118,102,118,122
471: |:2o4'g1b<d' r1r1  'd4f+a'a4'g4<ce''a4<df+':|  @u110'g1b<d'
472: r1r1  z105,100,110,105r2'd+f+b8.''d+f+<c+8.''ab8<d+f+'
473: 'f1ab-<d'  r1
474: z105,+3,113,116,124,122,124,114,118,120,118,110,115,115
475: r8d8g8b8<d4.e8  f+2a2  >b-2<c2  >g8f+8e8t68¥50f+2&f+8&  f+1
476:
477: (t5)o3|:2|:4dbeg:||:4db-eg:||:4f+<d>ab:|
478: |:2f+<d>ab:|a<ecd>f+<d>ab:||:4dbeg:||:4db-eg:||:4f+<d>ab:|
479: |:4d+<c+>ab:||:4f<d>gb-:||:4g<d>ab:||:4dbeg:||:4daef+:|
480: |:2db-f+g:||:2e<c>gb-:|  d<d>f+ad<d>d8&¥50d2&  d1
481:
482: (t6)o1z110,95,110,110g8r8.ggrg8r4.  <c8r8.ccrc8r4.>
483: f+8r8.f+f+rf+8r4.  z110,95,110,95,110,100f+8r8.f+af+a4<d4>
484: z110,95,110,110g8r8.ggrg8r4.  <c8r8.ccrc8r4.>
485: f+8r8.f+f+rf+8r4.  z95,110,110,100r4f+8e8a4d4
486: z110,95,110,110g8r8.ggrg8r4.  <c8r8.ccrc8r4.>
487: f+8r8.f+f+rf+8r4.  z110,95,110,110,118,105
488: b8r8.bbrb8.<f+8.b8>  z110,95,110,110b-8r8.b-b-rb-8r4.
489: @u110a8r8<@u118@b-3415,0,1a*16&@b0a*128  z110,95,110,110
490: d8r8.ddrd8r4.  z110,95,110,115,105d8r8.ddra4<d4>>
491: z105,110,118,100,95,110,110b-2<c2  d8>a8e8d8&¥50d2&  d1
492:
493: (t8)|:2o5z110,118,102,118,122d1  r1r1  d4>a4<e4f+4:|
494: o5@u110d1>  r1r1  z115,110,120r2b8.<c+8.f+8  r1o4r1
495: z105,108,113,116,124,122,124,114,118,100,98,90,95,95
496: r8d8g8b8<d4.e8  f+2a2  >b-2<c2  >g8f+8e8¥50f+2&f+8&  f+1
497:
498: (t10)z110,110,115,95,110,100o3a4>'df+'a+f+8f+4'df+4'
499: |:6f+4'df+'a+f+8f+4'df+4':|
500: f+4'df+'a+f+8f+4z108,+6,106<cd>g8  z110,110,115,95,110,100
501: <a4>'df+'a+f+8f+4'c+f+4'  @u110f+8.<d>'df4'f+4'df+4'
502: z110,110,115,95,110,100f+4'df+'a+f+8f+4'df+4'
503: f+8.<c>'df4'<a8.g8.g8  z108,100,115,110,110,95,100,105,102
504: a4>'c+f+'a+f+8f+8f+8d<dc8>  z100,110,108,100d*2d*46<a2>g4
505: z110,100,115,110,110,100<a4>'c+f+'a+f+8f+4'c+4f+'
506: f+4'c+f+'a+f+8f+4'c+4f+'  z110,100,115,110,110,95,95,100
507: f+4'c+f+'a+f+8f+8f+f+<c4  z110,100,105,108a8>a+8d8<g8^2
508:
509: (t11)c2c2  |:10c4.c8c2:|  c4.c8c8.c8.c8  c4.c8c4..c
510: r4c2.  |:3c4.c8c2:|  c4.c8r2
511:
512: (p)
```

リスト2 PURE GREEN用カウンタ表示

```
 1:00004EC0 00000000   2:00004140 00000000   3:00004380 00000000   4:00004F80 00000000
 5;00004F80 00000000   6:00004F80 00000000   7:00000D80 00000000   8:00004F80 00000000
 9:00004380 00000000  10:00004EC0 00000000  11:00004EC0 00000000  12:000009C0 00000000
```

リスト3 Ridge racer(POWER REMIX)

```
 1: /=================================================/
 2: /                                                 /
 3: /    Ridge racer(POWER REMIX)   /
 4: /                          from RIDGE RACER  (c)NAMCO    /
 5: /                  --- for SC-55 Series ---               /
 6: /                     Composed  by S.Hosoe                /
 7: /                     Programed by K.Terada               /
 8: /                                                 /
 9: /=================================================/
10: .comment Ridge racer(POWER REMIX) Programed by K.Terada
11:
12: /== Intialize ======================================
13: (i)     / Initialize
14: (b1)    / Base Channel => MIDI
15: (o140)
16: (m1 ,3000)(amidi 1,1)
17: (m2 ,3000)(amidi 2,2)
18: (m3 ,3000)(amidi 3,3)
19: (m4 ,3000)(amidi 4,4)
20: (m5 ,3000)(amidi 5,5)
21: (m6 ,3000)(amidi 6,6)
22: (m7 ,3000)(amidi 7,7)
23: (m8 ,3000)(amidi 8,8)
24: (m9 ,3000)(amidi 9,9)
25: (m10,3000)(amidi10,10)
26: (m11,3000)(amidi10,11)
27: (m12,3000)(amidi10,12)
28:
29: .sc55_init 16       / SC-55 Initialize
30: .sc55_chorus 16{6} / Chorus => S.Delay
31: .sc55_v_reserve 16{2,2,2,4,3,4,1,1,1,4,0,0,0,0,0,0}
32:
33: /== Setting ========================================
34: /             ID        B.R  Bend    Vol    Mod
35: (t1)  @i$41,$10,$42 @g12 @b20    @v90   @m90 @h32
36: (t2)  @i$41,$10,$42 @g12 @b-20   @v90   @m90 @h32
37: (t3)  @i$41,$10,$42 @g12 @b0     @v90   @m90 @h32
38: (t4)  @i$41,$10,$42 @g12 @b0     @v100
39: (t5)  @i$41,$10,$42 @g12 @b0     @v100
40: (t6)  @i$41,$10,$42 @g12 @b0     @v100
41: (t7)  @i$41,$10,$42 @g12 @b0     @v100
42: (t8)  @i$41,$10,$42 @g12 @b0     @v127
43: (t9)  @i$41,$10,$42 @g12 @b0     @v100
44: (t10) @i$41,$10,$42 @g12 @b2000 @v110
45: (t12) {k.sign +f,+g,+a} / for Hi-Hat
46:
47: /== Main ===========================================
48:
49: /== Melody 1 =======================================
50: (t1) i0@83 @p32 @u100 @e100,80 o4 q8 l16 r1
51: (t1) |:8r1:|
52: (t1) [do]
53: (t1) |: |:<c4>a8.b8.r8r2<c8.>b8r4..
54: (t1) <c4>a8.b8.r8r2<c8e|d&d4>defg:|<d&d2 | |:16r1:| :|
55: (t1) @81          @u90  r1r1r1r2.>fgab<c1>g1<d1&d1
56: (t1) @82          @u100 @e80,60 l16
57: (t1) f8e-rfe-rc8dr>b-<cr>a8b-<cr>a8g8fe-fe-dcdc>b- l8
58: (t1) a.<d.ag.f.b-a.g.<c>b-.<e-.d f.f.f&f2&f1&_20f1&_f1
59: (t1) @v90 l16
60: (t1) |:r<c>rb-|_40:|~r2rab-<c>b-agfe-ggf8c>b-8
61: (t1) rab-a<fe-rdre-ddcdc>b-a8.a8.g8<d8.d8.c8>
62: (t1) a8.a8.g8a2&a1&_20a1&_a1~40
63: (t1) fgab-r8fgab-r8fgab-fgab-<c>rfgab-<d>rb-<cde-
64: (t1) fe-de-cd>b-<cd8.e-8.d8r|:_30d8r:|rr2 ~60
65: (t1) >rb-<cde-dc>ar<f>ra<f>rar
66: (t1) rb-r<g>rb-<g>rarar<f>ar<f>
67: (t1) b-r<g>rb-<g>raar<f>ra<f>ra
68: (t1) b-r<g>b-r<g>rb-<cracrarc
69: (t1) i8@18 @v70 @u120 @e60,80 o5
70: (t1) e1&e4.efg8.f8.c8d1&d2.>ab<cd
71: (t1) e1&e4.efg8.f8.c&{cdefga}
72: (t1) b8.g8.d8&d2&d2.>{ab<cdefgab}4
73: (t1) <c8.>b8.a8<c8.>b8.a8
74: (t1) <c8.d8.c8c8.>b8.a8b8.b8.g2&g1ed
75: (t1) _30e2.&|:4e16&~15:|er8. _60|:6e8&~5:| @v70
76: (t1) _30d2.&|:4d16&~15:|dr8. _70|:4d8&~5:| @v70
77: (t1) {>>efgab<cdefgab<cd}4
78: (t1)        @v90 @u115 @e90,80
79: (t1) <e8re-8rd8rc8rcr>drr2.<c>b8.
80: (t1) r2e8.a8.b8<|:8c32>a32<:||:8d32>b32<:|
81: (t1) e8.g8.e8|:8ge:||:8g32e32:|b8.g8.d8c4d4
82: (t1) c8.c8.c8>b8r4.<
83: (t1) i0@82        @u100 @e127,40 o5
84: (t1) c1&c2..dc&c8.>a8.g8<e4d1&d8>fg<
85: (t1) |:e8.d8.c8:|g8.a8.g2f8g1&g1
86: (t1) @83   @p32 @u100 @e100,80  o4
87: (t1) |:7r1:|r2.defg
88: (t1) [loop]
89:
90: /== Melody 2 (Delay) ===============================
91: (t2) i0@83 @p80 @u70 @e100,80 o4 q8 l16 r1 r8
92: (t2) |:8r1:|
93: (t2) [do]
94: (t2) |: |:<c4>a8.b8.r8r2<c8.>b8r4..
```

```
 95: (t2) <c4>a8.b8.r8r2<c8e|d&d4>defg:|<d&d2 | |:16r1:| :|
 96: (t2) @81 @p96 @u80 r8 r1r1r1r2.>fgab<c1>g1<d1&d2..
 97: (t2) @82 @p64 @u60 @e100,40 l16
 98: (t2) f8e-rfe-rc8dr>b-<cr>a8b-<cr>a8g8fe-fe-dcdc>b- l8
 99: (t2) a.<d.ag.f.b-a.g.<c>b-.<e-.d f.f.f&f2&f1&_20f1&_f1
100: (t2) @v90 l16
101: (t2) |:r<c>rb-|_40:|~r2rab-<c>b-agfe-ggf8c>b-8
102: (t2) rab-a<fe-rdre-ddcdc>b-a8.a8.g8<d8.d8.c8>
103: (t2) a8.a8.g8a2&a1&_20a1&_a1~40
104: (t2) fgab-r8fgab-r8fgab-fgab-<c>rfgab-<d>rb-<cde-
105: (t2) fe-de-cd>b-<cd8.e-8.d8r|:_30d8r:|rr2 ~60
106: (t2) >rb-<cde-dc>ar<f>ra<f>rar
107: (t2) rb-r<g>rb-<g>rarar<f>ar<f>
108: (t2) b-r<g>rb-<g>raar<f>ra<f>ra
109: (t2) b-r<g>b-r<g>rb-<cracrarc
110: (t2) i8@18 @v70 @u110 @e60,80 o5
111: (t2) r2..|:7r1:|
112: (t2) |:4f8.f8.f8:|g8.g8.d2&d1r8
113: (t2) |:4r1:|
114: (t2)       @v90 @u110 @e80,40
115: (t2) <r1eb32<c.>eb8e'a8f'e'g8g'r4r1r1
116: (t2) c1&c1d8.>b8.g8g4g8>{gab<cdef}8<
117: (t2) g8.g8.>>{gab<cdef}8<
118: (t2) g8>{fedc>bagfedc>ba}8{gab<cdefgab<cdefg}4
119: (t2) i0@53        @u90  @e40,40 o4 r8
120: (t2) e1&e2..fe&e8.c8.>a8<g4>b1&b8ab
121: (t2) |:<c8.>b8.a8:|<e8.f8.e2d8e1&e1
122: (t2) @83   @p80  @u70  @e100,80 o4
123: (t2) |:7r1:|r2.defg
124: (t2) [loop]
125:
126: /== Melody 3 ======================================
127: (t3) i0@86 @p64 @u70  @e100,127 o4 q8 l16 r1
128: (t3) |:8r1:|
129: (t3) [do]
130: (t3) |: |:a4f8.g8.r8r2a8.g8r4..
131: (t3) a4f8.g8.r8r2a8<c>|b&b4defg:|b&b2 | |:16r1:| :|
132: (t3) @53          @u100
133: (t3) a2.&a16..a-16a16b-4a4e4f4g2..e8c2.cde-f
134: (t3) g1g8.a-8.g8b-8.a-8.f8g1&@m800g1@m90 r64
135: (t3) @82   @p96 @u100 @e80,60 l16
136: (t3) <c8>b-rab-rg8arfgrc8garf8e-8dcdc>b-ab-ag l8
137: (t3) f.b-.<fe-.d.gf.e-.ag.<c.>b-<
138: (t3) @b-683,120,0 d&d16 d&d16 d&d2&d1&_20d1&_d1~40 @b0
139: (t3) l16 |:rgrf|_40:|~r2rfgagfe-dce-e-d8>ag8
140: (t3) rfgf<dc>rb-r<c>b-b-ab-agf8.f8.e-8b-8.b-8.a8
141: (t3) f8.f8.e-8f2&f1&_20f1&_f1~40
142: (t3) defgr8defgr8defgdefgardefgb-rgab-<c
143: (t3) rc>b-<c>ab-gab-8.<c8.>b-8r|:_30b-8r:|rr2 ~60
144: (t3) rgab-<c>b-afr<d>rf<d>rfr
145: (t3) rgr<e->rg<e->rgrfr<d>fr<d>
146: (t3) gr<e->rg<e->rgfr<d>rf<d>rf
147: (t3) gr<e->gr<e->rgar<f>ar<f>ra
148: (t3) @18 @v70 @u80  @e60,80 o5
149: (t3) c1&c4.ccc8.c8.>a8b1&b1<c1&c4.ccc8.c8.>a&{ab<cdef}
150: (t3) g8.g8.>b8&b2&b2.{fgab<cdefg}4d8.d8.d8d8.d8.d8
151: (t3) d8.d8.d8d8.d8.d&{efgab}d8&{efgab}d8.>b2&b1r8<
152: (t3) _30c2.&|:4c16&~15:|cr8. _60|:6c8&~5:| @v70 >
153: (t3) _30b2.&|:4b16&~15:|br8. _70|:4b8&~5:| @v70
154: (t3) {>>cdefgab<cdefgab<}4
155: (t3) i8  @v90 @u90  @e80,40
156: (t3) <b32<c.>f+32g32b8gb8ga8gb-gbg
157: (t3) g+32a.rg8rgref+32g32rg<dc8.>
158: (t3) a64b-64b.b-b8f+g8c8.f8.g8|:8a32r32:||:8b32r32:|
159: (t3) <c1>g+1<g8.d8.>b8a4b8{defgab<c}8>
160: (t3) g8.g8.{defgab<c}8>
161: (t3) g8{dc>bagfedc>bagf}8{efgab<cdefgab<cde}4
162: (t3) i0@53     @u110 @e60,60 o4
163: (t3) e1&e2..fe&e8.c8.>a8<g4>b1&b8ab
164: (t3) |:<c8.>b8.a8:|<e8.f8.e2d8e1&e1
165: (t3) @86 @p64 @u70  @e100,127 o4
166: (t3) |:7r1:|r2.defg
167: (t3) [loop]
168:
169: /== Synth Strings & Chord 1 (Left) ====================
170: (t4) i127@49 @p80 @u110 @e100,60 o5 q7 l16 r1
171: (t4) |:2r8'dfa<c>'r8'dfa<c>'r8
172: (t4) r1r'cdfg'r4.r8'dfab'rr2.r1:|
173: (t4) [do]
174: (t4) |:8r8'dfa<c>'r8'dfa<c>'r8
175: (t4) r1r'cdfg'r4.r8'dfab'rr2.r1:|
176: (t4) i0@5    @p16 @u80 @e127,40 o4 q8
177: (t4) d1d8.d8.d8>a4a4<c1e1c1|:c8.c8.c8:|d1&d1
178: (t4) @27             @u70 @e20,20  o4
179: (t4) |:6re-r8e-rcre-r8e-r4 rcr8cr>b-rb-r8b-r4
180: (t4)   rb-r8b-rb-rb-r8b-r4<rcr8cr>b-rb-r8b-r4:|
181: (t4) @6              @u90 @e40,40  o4
182: (t4) |:6 'e8.fa'&_30'e8.fa'&~'e8fa'&'e2fa'&'e1fa'
183: (t4)     'd8.fg'&_30'd8.fg'&~'d8fg'&'d2fg'&'d1fg' :|
184: (t4) i127@49 @p80 @u110 @e100,60 o5 q7
185: (t4) |:4 r8'dfa<c>'r8'dfa<c>'r8
186: (t4) r1r'cdfg'r4.r8'dfab'rr2.r1:|
187: (t4) [loop]
188:
189: /== Organ & Chord 2 (Center) ==========================
190: (t5) i127@9 @p48 @u100 @e00,30 o3 q6 l16 r1
191: (t5) |:4 d8'fa<c>'rd'fa<c>'r'dgb'd'dgb'rd'dgb'd'c8eg'
192: (t5) d8'cfa'rd'cfa'r'deg'd'deg'rd'deg'd'f8gb':|
193: (t5) [do]
194: (t5) |:16d8'fa<c>'rd'fa<c>'r'dgb'd'dgb'rd'dgb'd'c8eg'
195: (t5) d8'cfa'rd'cfa'r'deg'd'deg'rd'deg'd'f8gb':|
196: (t5) i0@5   @p64 @u80 @e127,40 o4 q8
197: (t5) c1c8.c8.c8>g4g4b1<c1>b-1|:b-8.b-8.b-8:|b1&b1
198: (t5) @27             @u70  @e20,20 o4
199: (t5) |:6rb-r8b-rgrar8ar4 rgr8gre-rfr8fr4
200: (t5)    re-r8e-rdrfr8fr4 rgr8gre-rfr8fr4:|
201: (t5) @6              @u100 @e40,40 o4
202: (t5) |:2r4.a8&a2&a1 | r4.g8&g2&g1:| r4.g4.&'g4d'&'g1d'
203: (t5) |:4r4.a8&a2&a1   r4.g8&g2&g1:|
204: (t5) i127@9 @p48 @u100 @e00,30 o3 q6
205: (t5) |:8d8'fa<c>'rd'fa<c>'r'dgb'd'dgb'rd'dgb'd'c8eg'
206: (t5) | d8'cfa'rd'cfa'r'deg'd'deg'rd'deg'd'f8gb':|
207: (t5) 'd8gb'r8r2.
208: (t5) [loop]
209:
210: /== Effects & Chord 3 (Right) =========================
211: (t6) i0@102 @p48 @u90 @e127,20 o5 q8 l16 r1
212: (t6) |:'f1<c>'r1'f1b'r1:|
213: (t6) [do]
214: (t6) |:8 r1:|
215: (t6) @102   @p48 @u90 @e127,20
216: (t6) |:'f1<c>'r1'f1b'r1:|
217: (t6) |:16r1:|
218: (t6) @5    @p111 @u80 @e127,40 o4 q8
219: (t6) f1f8.f8.f8c4d4e1g1e-1|:e-8.e-8.e-8:|e1&e1
220: (t6) @27             @u70 @e20,20  o4
221: (t6) |:6rfr8fre-rfr8fr4 re-r8e-rcrdr8dr4
222: (t6)    rcr8crcrdr8dr4 re-r8e-rcrdr8dr4:|
223: (t6) @6              @u90 @e40,40  o4
224: (t6) |:6_30'e8.fa'&~'e8.fa'&_'e8fa'&'e2fa'&~'e1fa'
225: (t6)     _30'd8.fg'&~'d8.fg'&_'d8fg'&'d2fg'&~'d1fg':|
226: (t6) |:16r1:|
227: (t6) [loop]
228:
229: /== D.Guitar & Mute Guitar ============================
230: (t7) i0@31 @p80 @u100 @e60,80 o2 q8 l8 r1
231: (t7) |:4dfga16d.fgadfgf16<c.>bgf:|
232: (t7) [do]
233: (t7) @37   @p64        @e00,40 o4 l16
234: (t7) |: drrdr2.ddrdr2.ddfdr2.ddrdr2.:|
235: (t7) @31   @p80        @e60,80 o2 l8
236: (t7) |:4dfga16d.fgadfgf16<c.>bgf:|
237: (t7) @37   @p64        @e00,40 o4 l16
238: (t7) |:4drrdr2.ddrdr2.ddfdr2.ddrdr2.:|
239: (t7)       @p24 @u90  @e60,60 q7
240: (t7) |:|:3d_40dd~|dr8:|c+:| |:|:3c_cc~c|r8:|:|
241: (t7) |:|:3e-_e-e-~e-|r8:|:| |:|:3d_dd~d|r8:|:| >
242: (t7)       @p64        @e00,40 q6
243: (t7) |:6<c8rc8r>b-r<crc8r8>c8 b-8rb-8rgrara8r8>a8<
244: (t7)     g8rg8r g r ara8r8>a8<b-8rb-8rgrara8r8>a8<:|
245: (t7) x$40,$17,$1c,$00 / for Panpot => Random
246: (t7)                   @e40,90
247: (t7) aaaargr8r4.aaa_aa~ar2r8_|:4g32~10:|
248: (t7) g_40gg~gr2.ggggr2.aaaar2.rarar2.g_gg~gr2.rgrgr2.
249: (t7) aaaar2.rarar2.|:g_gg~gr2.:|
250: (t7) aaaaararr4.gra_a~arr2.g_gg~gr2.gg_g~gr2.
251: (t7) arrar2.|:4a32:|rar2.g_gg~gr2.|:5r1:| <
252: (t7)       @p80 @u100 q7
253: (t7) |:4 |:d_dd~dr8d_dd~dr8d_dd~|1e:||2c:| :|
254: (t7) |:8r1:| q8
255: (t7) [loop]
256:
257: /== Bass ==============================================
258: (t8) i0@36 @p64 @u100 @e20,20 o1 q8 l8 r2(e-2>d),24
259: (t8) @38           @u110 @e40,25 o1
260: (t8) @y$01,$20,$2a / for Cutoff Freq. => -22
261: /      ↑音を落ち着かせるためです   アマリコウカナイカナ (・・;)
262: (t8) |:4 dfga16dd16fgadfgf16<cc16>bgf:|
263: (t8) [do]
264: (t8) |:16dfga16dd16fgadfgf16<cc16>bgf:| u+10 @e60,30
265: (t8) d.d16r2r<d>d.d16r(<d2>d),24r
266: (t8) c.c16r2r<c>c.c16r2r<c>
267: (t8) a-.a-16r2r<a->a-.a-16r4(<c4.>c),24<a->
268: (t8) g.g16r2r<g>g.g16r4g16g8.<c16d16>g16<g16> @e40,25
269: (t8) |:6ce-fg16cc16e-fge-fb-g16<cc16>b-fg
270: (t8) gb-<c>b-16gg16fe-fe-e-fe-16gg16fe-d:|
271: (t8) |:6|:efga16ee16fga:||:defg16dd16efg:|:|
272: (t8) |:8dfga16dd16fga|dfgf16<cc16>bgf:|drr2. u
273: (t8) [loop]
274:
275: /== Slap Bass =========================================
276: (t9) i0@37 @p48 @u100 @e20,20 o1 q7 l8 r1
277: (t9) |:4d16<~25d16_>fga16dd16fga
278: (t9)    d16<~d16_>fgf16<cc16>bgf:|
279: (t9) [do]
280: (t9) |:8d16<~d16_>fga16dd16fga
281: (t9)    d16<~d16_>fgf16<cc16>bgf:|
282: (t9) |:8r1:|
283: (t9) |:4d16<~d16_>fga16dd16fga
284: (t9)    d16<~d16_>fgf16<cc16>bgf:|
285: (t9) |:7r1:|r2.~<c16d16>g16<g16>_
286: (t9) |:24r1:|
287: (t9) |:4 |:e16<~e16_>fga16ee16fga:|
288: (t9)      |:d16<~d16_>efg16dd16efg:| :|
289: (t9) |:e16<~e16_>fga16ee16fga:|
290: (t9)   d16<~d16_>d-c>bb-aa-g
291: (t9)  <d16<~d16_>>g-ga-ab-b<c
292: (t9) |:e16<~e16_>fga16ee16fga:|
293: (t9) |:d16<~d16_>efg16dd16efg:|
294: (t9) |:16r1:|                        / ~ と _ に注意
295: (t9) [loop]
296:
297: /== Snare Drum ========================================
298: (t10) i0@1 @p64 @u30 @e60,20 o2 q8 l16
```

```
299: (t10) |:4u+10du-15du+5ddu+25:| @u100 l8
300: (t10) |:15rd:|d16d16d16d16
301: (t10) |:12rd:|d16ddd16dddd16d16d16d16
302: (t10) [do]
303: (t10) |:|:15rd:|d16d16d16d16
304: (t10) |:12rd:|d16ddd16dddd16d16d16d16:|
305: (t10) |:28rd:|l16u-20
306: (t10) |:u+30du-30ddd:||:u+30du-30d:|u+30du-20ddd l8 u
307: (t10) |:15rd:|d16d16d16d16
308: (t10) |:12rd:|d16dddd16ddd16d16d16d16
309: (t10) |:6 r4d4:|r4d16d16r4.d4|:7r4d4:|r4d16u-20d16d u
310: (t10) |:15r4d4:|r4d16d16rr.d16r4
311: (t10) |:14r4d4:|r.u-10d u d16d16d16
312: (t10) |:10r4d4:|r16d16d16r16drru-10d u drrdr2d4
313: (t10) l16u+10du-10ddddu+10dr8d4|:u+10du-10d:| u
314: (t10) |:4 |:3r4d4:|r4d8 u-40|:4d32:| u r4d4r4d4
315: (t10) |1 u-50|:8d32:|u|:4du-20:|r8|:4d32u+20:|uddrd :|
316: (t10) |2 ddddd4r8.u-40|:d32u+20:|ddrd :|
317: (t10) |3 u-50|:8d32:|u|:4du-20:|r8|:4d32u+20:|uddrd :|
318: (t10) |4 rdddrdddrd32d32dr du-20dd u d:|
319: (t10) |:15r8d8:|dddd|:12r8d8:|dd8d8dd8d8d8dddd
320: (t10) |:4 |:3r4d4:|r4d8 u-40|:4d32:| u r4d4r4d4
321: (t10) |1 u-50|:8d32:|u|:4du-20:|r8|:4d32u+20:|uddrd :|
322: (t10) |2 ddddd4r8.u-40|:d32u+20:|ddrd :|
323: (t10) |3 u-50|:8d32:|u|:4du-20:|r8|:4d32u+20:|uddrd :|
324: (t10) |4 d4r4 rdrd32d32ddrd:| l8
325: (t10) [loop]
326:
327: /== Bass Drum ======================================
328: (t11) @u127 o1 q8 l8 r1
329: (t11)    |:7brbr16bb16rbr:|bbb.bbr16br
330: (t11) [do]
331: (t11) |:|:7brbr16bb16rbr:|bbb.bbr16br:|
332: (t11) |:15brbr16bb16rbr:|bbb.b16bbbb
333: (t11) |:4b.b16r4..b16rbb.b16r2rb:|
334: (t11) |:8brr.b.brb:|r16b.r.b.brb |:brr.b.brb:|
335: (t11) brr.b4b16rb b.b16r4bbrb|:3brr.b.brb:|
336: (t11) bbr4bbrb br4.bbrb bbr4bbrb brr.b.brb
337: (t11) |:4brr.bbb16|rb:|b16b16b16r16
338: (t11) |: |:7br4.bbrb:| |1b8.b16r4bbrb:| |2b4b4bbrb:|
339: (t11) |:7brbr16bb16rbr:|bbb.bbr16br
340: (t11) |: |:7br4.bbrb:| |1b8.b16r4bbrb:| |2b4r4bbrb:|
341: (t11) [loop]
342:
343: /== Hi-Hat & Cymbal =================================
344: (t12) @u100 o2 q8 l8 r1
345: (t12) |:2<c+*0>|:6fg:|fafg|:4fg:|agfgfafg:|
346: (t12) [do]
347: (t12) |:4<c+*0>|:6fg:|fafg|:4fg:|agfgfafg:|
348: (t12) |:<c+*0>|:16a4:|:|
349: (t12) |:2<c+*0>|:6fg:|fafg|:4fg:|agfg | fafg:| a2 l16
350: (t12) <c+*0>a8gg|:fggg:|fgagfgf32f32gfaggfgggfgag
351: (t12) |:3fggg:|fgagfagg|:fggg:|'<g!>f'ggg
352: (t12) |:3fggg:|fgagfgf32f32gfaggfgggfgag
353: (t12) fagg|:fggg:|fgagagggfaggfgggaggg
354: (t12) <c+*0>|:3fgfgfgffgffgfgag:|
355: (t12) |:fgfgfgffgaagfgfg|<c+*0>:|
356: (t12) |:3fgfgfgffgffgfgag:|fag'<c+>g'fgffgffgfgag
357: (t12) |: fgfgfgffgffgfgag:|
358: (t12) a8.a8.f<c+>a8.a8.fra8.a8.af<c+>ffgfgag
359: (t12) |:3fgfgfgffgffgfgag:|
360: (t12) '<c+>a8'fgfga8.gfga8fg a8.agra8.agra8gr
361: (t12) a4.aaa8.<g!8.>'<c+>f'g
362: (t12) fgfgfgffgffgfgag fgfgfgfagaaafgfg
363: (t12) |:3agagu+27au-27gfagaaa
364: (t12) |1r8fg:| |2<g!>rfg:| |3r4:|
365: (t12) |:2<c+*0>|:3fga8fgffgffgfgag fgfgfgffgffgfgag:|
366: (t12) fga8fgffgffgfgag |1a4fgffgffgfgag:|
367: (t12) |2a4a4r4.a8:| l8
368: (t12) |:2<c+*0>|:6fg:|fa|:5fg:|agfgfa4.:| l16
369: (t12) |:2<c+*0>|:3fga8fgffgffgfgag fgfgfgffgffgfgag:|
370: (t12) fga8fgffgffgfgag |1a4fgffgffgfgag:|
371: (t12) |2a8g8r4'<c+4>a'fgag l8
372: (t12) [loop]
373:
374: (p) / 横浜国大吹奏楽団 入団者募集中!! (^^;)
```

リスト4 Ridge racer(POWER REMIX)用カウンタ表示

```
 1:000006C0 00004E00   2:000006D8 00004E00   3:000006C0 00004E00   4:000006C0 00004E00
 5:000006C0 00004E00   6:000006C0 00004E00   7:000006C0 00004E00   8:000006C0 00004E00
 9:000006C0 00004E00  10:000006C0 00004E00  11:000006C0 00004E00  12:000006C0 00004E00
```

(進)の「ちょっといいですかぁ?」

◆PURE GREEN

このこだわりには圧倒されます。「コピーはここまでやらなくては」と訴えかけられている気もします。努力と試行錯誤の賜でしょう。

注目したいのがZコマンドによるきめ細かいベロシティ設定。楽器経験者でも難しくて面倒な作業ですが,自然にサラリとこなしてあって作者の熟練度を感じます。ご苦労さま。

さらに上を目指すなら,発音タイミングに凝ることですね。ステップを前後にズラすことで,ソロ演奏では流れをさらに人間クサく,リズム全体に使えばバンド特有のクセなんかも再現できます。ただ,これもかなりの研究が必要になりそうですが……。

さて,アラ捜しというつもりはないですが,気になった点を少々。

ストリングスがきついです。コーラスが強いのか,余計なうねりを感じるのも気になります。また,トラック8でオクターブ下の音を重ねていますが,これはホルン系の音がいいですね。

エフェクトはうまいですが,私の好みでいうと,リバーブはHall1でちょっとゲート効果っぽくしてもいいんじゃないかと思います。

それから,トラック5のマリンバ。原曲では深くエフェクトがかかっていて,どこから鳴っているか摑めないですよね。手に取るようにわかる音もあり,遠くに感じる音もある。まあ,ここまで再現できれば文句はないですね。「空間の設定」は難しい……。

余計なことですが,音程がCDと違う場所があるので書いておきます。トラック4,8で
d8e8g8<d4.e8 f+2g4a4 が
d8g8b8<d4.e8 f+2g4a4
実際はこのようになっています(音楽的には気になりませんが)。

◆Ridge racer

「完全再現より,曲として成り立つように作った」という選択は正しいと思います。難しい曲にもかかわらず,重要な音採りで致命的な間違いがないのがいい。音楽の経験が活きていますね。

ただ全体的に寂しいです。パーシャルの限界を考えた構成はうまいと思いますが,最後の調整が足りないのか,中途半端な印象があります が,どうでしょうか。音色の選択を含めて,練り込んでみるといいかも。

ちょっとベースの音量が大きいですね。一貫してスラップで演奏していますが,ここは原曲と同じパターンのほうが私は好きです。加えて,ベースのエフェクタはやっぱり@e0しか考えられないでしょう(笑)。

カットオフ周波数を極端に下げてベースを太くするというテクニックは,SC-55では必須です。効果がないってことはありませんよ。

あとは,ステレオ効果をもっと有効に使うとさらによくなるかも。音と音との絡みやリズムの表と裏を,より一層濃く表現できますしね。

なんだか辛口評価になってしまいましたが,決して出来の悪い作品ではありません。気を悪くしないでくださいね。今後に期待します。

◆　◆　◆

さて,今月のローテク工作実験室に関しての個人的なコメント。

本体への入力でAD PCM音を作っている人にとっては,入力フィルタの改造は満足度120%といったところか。2個の電解コンデンサの入れ替えで,清く正しく低音が録れるようになります。単なるフィルタの調整ですが,この点に関してだけいえば「音質が向上した」とさえ感じます。

本体の改造を奨励する気はありませんが,非常にコストパフォーマンスの高いローテクです。私は感動しました。

出力フィルタ改造は,いろんな抵抗を何本か買って視聴しながら実行しましょう(ただし3本同じ値で)。カットオフ周波数が15kHz程度になると,いままで使っていた音がノイズだらけになるってこともあります。　(進藤慶到)

(善)のゲームミュージックでバビンチョ

西川善司

今年も恒例の「ゲームミュージックフェスティバル」が行われるぞ。開催日は7/30(土)，7/31(日)の2日間だ。場所は日本青年館。出演バンドは以下のとおり。

・7/30(土)
國府田マリ子&矩形波倶楽部 (KONAMI)
葉山宏治&ブラザーズ (NECアベニュー)
B-univ (SEGA)

・7/31(日)
ZUNTATA (TAITO)
GAMADELIC (DATA EAST)
新世界楽曲雑技団SPECIAL BAND (SNK)

なお，チケットはもう発売中なので買い忘れていた人はチケットぴあ(☎03-5237-9999)，チケットセゾン(☎03-5990-9999)へ走れ！ その他，GM事務局(☎03-3470-9858)，ニッポン放送ミュージックセンター(☎03-3213-4444)でも買えるよ。

＊　＊　＊

●ワールドヒーローズ2 JET
VHS:PCVP-11450　4,800円(税込)
LD:PCLP-00513　4,800円(税込)
ポニーキャニオン　発売中

どこまで真面目なのか，不真面目なのかまったく「謎」の，イロモノキャラ勢ぞろいの対戦格闘ゲーム「ワー・ヒー2 JET」(ADK)のビデオが発売された。前半は各登場キャラクターの必殺技/連続技をスローモーションを駆使して紹介。後半は主人公・半蔵のエンディングまでのダイジェストプレイを収録している。エンディングはゲームキャラの声優たちの「お笑い」罵り合い漫才で幕を閉じる。ゲームの攻略ビデオというよりはプロモーション的な内容だ。

お勧め度　7

●極上パロディウス
CD:KICA-7641　2,800円(税込)
キングレコード　発売中

久々のコナミのヒットシューティングゲーム「極上パロディウス」のオリジナルサウンドアルバム＋α。脳天気な敵キャラたちのアホ顔にはさらに磨きがかかり，マイキャラたちも極まった意味不明さを装って再登場。前作以上に賑やかで楽しい(変な？)魅力のあるゲームになっている。

さて，音楽のほうも前作以上に楽しいものばかり。世界中のクラシックから民謡，童謡までを矩形波倶楽部独自のセンスでリニューアルしている。モチーフが知名度の高いものなので，ゲームをまだ遊んだことがない人にもお勧め。でも，コレを聴けばきっとゲームのほうもプレイしたくなってくるはずだ。音楽がパワーアップしたことはもちろんだが，ハードウェアのほうもパワーアップしたせいか，音質も非常に完成度が高い。発音数や音色容量の制限から解放されたような，「ゲームミュージック」らしからぬ自由な編曲を感じることができる(全43曲)。

お勧め度　10

●ナムコ・ゲームサウンド・エクスプレス Vol.13
NEBULASRAY
CD:VICL-15027　1,500円(税込)
ビクターエンタテインメント　7/21発売

近ごろ大型筐体系のゲーム以外ではいまいちパッとしなかったナムコだったが，この「ネビュラスレイ」はシューティングゲームに新しい流れを引き起こす可能性をも秘めているといっても過言ではないほどの作品だ。ゲームに登場するキャラはすべて本格的に3Dモデリングを施し，これをスプライトパターンに落としたものを使っているらしく，2D縦スクロールシューティングの常識を超えた「動き」を見せてくれる。一度ご覧あれ。音楽のほうはプログレッシブなフュージョン系が中心。ちょっと前に流行ったギコガコベースが心地よいスペーシーサウンド満載といったところ。ライナーノーツの「演奏者：電脳演奏遊戯“苦亜吐露”」にはどんな秘密が？

お勧め度　8

●ミュージック フロム 英雄伝説Ⅲ
もうひとつの英雄たちの物語～白き魔女～
CD:KICA-1146～7　3,800円(税込)
キングレコード　7/21発売

もういまや「ドラゴンスレイヤー」のサブタイトルはどこかに消え去ってしまい，ファルコムの新たな代表的シリーズとしてひとり歩きを始めたこの「英雄伝説」シリーズ。このパートⅢのオリジナルサウンドアルバム＋αが発売される。2枚組のアルバムで全68曲，収録時間にして140分。オリジナルサウンドはPC-98版のFM3＋PSG3音という，PCMが当たり前の現在では時代錯誤としか思えない音源による演奏だが，曲は素晴らしい。メロディアスなものが多く，古きよき時代のゲームミュージックを彷彿とさせてくれる。しかしファルコムはそろそろ，PC-98のゲームも86音源ボード(FM6，PSG3，リズム6，PCM2)などの最新音源に対応させないと時代遅れになっちゃうよ。

お勧め度　8

●コナミ・パーフェクトセレクション ドラキュラ・バトル
CD:KICA-1145　3,000円(税込)
キングレコード　7/21発売

今回のオールアレンジアルバムの発売は，ファンからのリクエストのフィードバックによるものだという。コナミの「ドラキュラ」人気は相当なものだ。全曲ハードロック系のアレンジで，ドラキュラのアレンジものとしてはいちばんの出来だと思う。収録曲は定番「BEGINNING」「BLOODY TEARS」「VAMPIRE KILLER」を含む全10曲。ドラキュラファンなら買いだね。

お勧め度　9

極上パロディウス

ドラキュラ・バトル

第147部　シューティングゲーム作成講座(2)

●キャラクター管理とは？

複数のキャラクターを同時に動かす。

といっても，実際に動かしているのは逐次処理のプロセッサなので，結局はキャラクターをひとつずつ動かしているだけです。ひとつずつ動かしていても，その動作が非常に高速であるため，キャラクターが同時に動いているように見えるのです。

で，複数のキャラクターを同時に動かすいちばん簡単な方法は，それぞれ動かしたいキャラクターのルーチンを用意し，順番に呼び出してやれば実現できます。つまり，3個のキャラクターが登場するときには，それぞれ3個のメインルーチンを組んでやればいいのです。

当然のことながら，こんな方法でキャラクターを動かしているプログラムは，いまどきありません。なぜならば，ゲーム中に登場するキャラクターの種類，個数は不特定数だからです。それに，同じ種類のキャラクターを動かすためには，動かしたい個数分メインルーチンを用意しなくてはならないなんて，明らかに無駄というものです(処理時間もかかるしね)。

では，無駄と考えられている部分を省いて具体的に効率を上げることを考えます。

キャラクターを動かすためには，表示するための座標を格納するワークエリア，そのほかの動作用カウンタなどが必要となりますね。まず，これを共通化しましょう。

すると，その共通化したワークエリアというのは，一定のサイズになります（当然ですね)。すると，インデックスレジスタを使ってワークエリアをアクセスできるようになります。

1キャラクターあたりのワークサイズが10バイトで，キャラクター1(CHR1)を2個，キャラクター2,3(CHR2,3)を1個ずつ動かすならば，

```
CHR1:   DS  10
CHR2:   DS  10
CHR1:   DS  10
CHR3:   DS  10
```

という具合にワークエリアを用意すればいいわけです。

次に，キャラクターの種類ごとのメインルーチンは共通ですから，キャラクターの種類を判別するフラグを用意します。すると，メインルーチンを共通化することができます（もちろんメインルーチンを呼び出すときには，ワークのトップアドレスをメインルーチンへ渡してやるのを忘れないように)。

最後にキャラクターの種類を判別するフラグを，さらにワーク使用フラグとしてチェックすれば，不特定個数のキャラクターも管理できるようになります。こうして，非常にフレキシブルなキャラクター管理ルーチンの出来上がりです。

なんだか，あまりにも簡単に話が進んでいるため，だまされた気分でしょうが，基本はこんなものです。「YGCS」でも同じようなことをやっているので，「シューティングゲーム作成講座」をよく読んで理解してください。

●配布はサークルの作業能力に合わせる

さて，皆様にご協力いただいている「S-OSフリーソフト化計画」。しかし，配布に当たっては結構問題を抱えているのが現状です。なにしろ，S-OS対応機種といってもその機種は多岐に渡っています。さらに，サークルに依頼するとしても，個人でやっているかぎり配布能力にも限界があります。最近，〝L-os Angeles〟を投稿してくださった中川氏（サークルLovers主宰）が協力を申し出てくれました(非常にうれしい)。配布に当たっては，もっと多くのサークルの皆さんの協力が不可欠なのです。7月号でも募集したように，配布に協力してくださるサークルの方からの連絡をお待ちしています。

なお，〝L-os Angeles〟とは，S-OS上位互換のサブルーチンをもち，MS-DOSフォーマットをサポートしたX1/X1turbo/Z用のOSです。階層化ディレクトリのサポートや，外部プログラムでの拡張性を備えた本格的なDOSといえます。詳しいことについては，また次回にでも紹介させていただくことにしましょう。

1994■インデックス

全機種共通
S-OS"SWORD"要

シューティングゲーム作成講座(2)

Uesugi Yuya
上杉 悠也

今回は，初期設定に必要な各種ワーク，テーブルの詳細を解説します。「YGCS」システムを使うためには欠かせない知識ですので，リファレンスマニュアル片手に理解を深めてください。

今月は「YGCS基礎の基礎」です。とりあえずすべての基礎である，ワークとデータの構造から話に入っていくことにしましょう。これは基本的にリファレンスマニュアルで説明されていますが，どうもわかりにくいという意見も寄せられているようなので，もう一度詳しく取り上げることにしました。システムを使ううえで非常に重要な知識なので，必ず理解してください。すでに理解している人は，復習のつもりで読んでください。

キャラクターワークの構造

キャラクターワークとはなにか？　キャラクターワークとは正式（？）にはキャラクター管理ワークといい，文字どおりキャラクターを円滑に管理するために作られたワークのことを指します。

実際，キャラクターがヘコヘコ動きまわるゲームでは，ほぼすべてに存在している(必要とされている)と思って間違いないでしょう。

しかしです。よくよく考えてみると，次のような会話が交わされていそうです。

A「なにを管理するの？」
B「キャラクター」
A「そうでなくて……」
B「まあ，簡単にいうと座標とかね」
A「ふーん」

しかし，皆さんはこのように簡単に納得しないでください。わからないところはとことん追求していく姿勢が大切なのです。

とりあえず，なにをするのかまったくわからない人もじっくり読んでみてください。何度も読み返しているうちに，必ず道は開けます（本当か？）。

それでは本題に入ります。YGCSでのキャラクターワークの構造と働きを，順を追って解説していきましょう。

リスト9（YGCSの標準ヘッダファイルです，打ち込んでおきましょう）と6月号に掲載されたサンプルプログラムを参照しながら読んでください。

1．FLG：フラグ

このフラグには2つの意味が定義してあります。

1つ目は，このワークが使用中かどうかの状態指定です。ここに0が書き込んであればワークは未使用，それ以外の値が書いてあれば使用中をそれぞれ表しています。

2つ目の意味は，キャラクター表示サイズの指定です。0以外の値が書いてあった場合(つまりワーク使用中の場合)，4種類のサイズ指定が可能です。各値の意味は，1から3までが固定サイズの指定で，4は可変サイズの指定です。

2．ATR：アトリビュート

このアトリビュートではヒットチェック用の属性を指定します。

まず最上位ビットの意味ですが，ここが0であればプレイヤー側の属性。逆に1であればそれ以外(敵など)の属性に判別されます。つまり，ヒットチェックのときにこのビットを見て，同じ属性のキャラクターどうしのヒット(衝突)を避けるわけです。

そのほかのビット（現在は0から2までが有効）は，プレイヤー側と敵側で意味が違ってきます。プレイヤー側は自分の種類(プレイヤー，オプション，弾の3種類)を表しています。敵側の意味は，プレイヤー側のどの種類のキャラクターとヒットチェックを行うかを指定します。

実際の例としては，

1）ヒット可能の組み合わせ
　　プレイヤー　敵本体
　　00000001_B - 10000111_B
　　オプション　敵弾
　　00000010_B - 10000011_B
2）ヒット不可能な組み合わせ
　　プレイヤー　プレイヤーの弾
　　00000001_B - 00000100_B
　　敵本体　敵弾
　　10000111_B - 10000011_B

となります。

簡単にいえば最上位ビットとは逆で，同じビットが立っている者どうしでヒットチェックが行われるわけです。

実際にはこの属性チェックを座標チェックの前に行います。簡単なプログラム例がリスト1ですので，読んでおいてください。しかし，YGCSを使うのであれば，判定はシステムが行ってくれます。

3．COD：キャラクターID

YGCSでは，各キャラクターのプログラムをID番号で管理していて，このワークはそのプログラムを指定するのに使います。

YGCSにおいてID番号とは，キャラクターテーブルに登録した順に0から割り振られるようになっています。そして，各プログラムの登録は各自で行ってもらうので，結果的にID番号は自由に設定可能なわけです。自分で管理しやすいように番号を設定しましょう。

実際のテーブルへの登録の仕方は，各種テーブル構造の項目で解説します。

4．CNT：カウンタ

このカウンタはキャラクターの出現と同

時にリセットされ，キャラクターのプログラムが呼び出されるたびにカウントアップされるようになってます。よって，このワークへの書き込みは基本的に禁止です（わかっていて書き込むなら問題なし)。参照のみで利用してください。

具体的な利用法については，最もオーソドックスな使い方としてタイミング取りに使う方法があります。つまり，キャラクターが出現してから，一定時間後にキャラクターモードを切り替えたりするときのチェックカウンタとして使うのです。ほかにもいろいろ利用方法はありますが，各自で工夫して使ってみてください。

5．XPS/YPS：XY座標

キャラクター管理ワークといって最初に思いつくのがこのワークです。最もメジャーなワークといっても過言ではありません。

YGCSでの座標管理は各座標ごとに2バイトで行います。1バイト目は直接画面の座標になりますが，2バイト目は座標計算用の小数部になっています。この小数に関する説明は前回の講座の「移動量の秘密」を参照してください。

この小数部の利用方法については，これからの講座の中で説明していきます。

6．XDF/YDF：XY移動量

このワークで扱う移動量は，直接いじる必要はありません。一部の決まったコールで使うだけです。そのコールというのはSIN関数移動関係と，加減速移動関係のコールです。これらのコールが自動的にこのワークを利用します。ただしこのワークはひと組しかないので，これらのコールを同時に使用することはできません。さらにシステムが使うという理由から，ここへの書き込みは極力避けてください。

7．PAT：パターンデータアドレス

ここには表示用パターンデータの格納アドレスを書いておきます。ただし，最初に説明したフラグ（FLG）が01_H（キャラクターサイズが1×1）だった場合は，直接キャラクタコードを書き込んでください。

8．MOD：キャラクターモード

このワークではキャラクターの動作モードを指定します。ただしこのワークが有効になるのは，サービスコールのMODCを利用した場合のみです。そのほかの場合，このワークはまったく無効になるので，自由に使用できるようになります。

このワークの使用方法については，リスト2にサンプルを示しておきます。

実際は，ガイドラインで推奨されているプログラムの組み方をしない場合には，必要ないものです。

9．DIR：方向

ここはSIN関数移動関係と，方向サーチのコールで参照しています。よって，初期化などはユーザーがすることになります。

YGCSでの方向制御は基本的に64方向です。それ以上の値をセットしても64以下に補正されますので，安心して（?）使用してください。

YGCSでは基本的にSIN関数移動がメインになると思われるので，このワークは頻繁に使うことになるでしょう。

使い方のサンプルをリスト3へ示しておきます。

10．SPD：スピード

ここも方向ワークと同じで，SIN関数移動から参照されます。

速度は10_Hで毎回1キャラクタ移動で，単

リスト1　衝突判定

```
 1: ;
 2: ;          衝突判定プログラムサンプル（一部）
 3: ;
 4: ;          IN D : ATR1
 5: ;             E : ATR2
 6:
 7:            LD       A,D
 8:            OR       A
 9:            JR       M,RETURN          ; プレイヤー側からのみ判定
10:
11:            LD       A,E
12:            OR       A
13:            JR       P,RETURN          ; プレイヤー属性どうしの判定はしない
14:
15:            AND      D
16:            AND      0EFH              ; 最上位（属性）ビットを捨てる
17:            JR       Z,RETURN          ; 衝突判定しない
18:
19: ;  ここまでくれば、衝突可能なキャラクターどうしであることがわかる
20: ;  これ以降で実際の衝突判定を行う
21:
```

リスト2　キャラクターモード使用例

```
 1: ;
 2: ;キャラクターモード（MOD）の使用サンプル（一部）
 3: ;  最初にキャラクターのプログラムが呼び出されたとき、
 4: ;       MODは0に初期化されている
 5: ;
 6: TEST_CHR:
 7:         LD      HL,TEST_TBL
 8:         JP      MODC_           ; MODの値によりジャンプ先を決定
 9: TEST_TBL:
10:         DW      TEST_MODE_0     ; 0 : INIT
11:         DW      TEST_MODE_1     ; 1 : MAIN
12: ;
13: ;       <<< TEST INIT
14: ;
15: TEST_MODE_0:
16:         ;
17:         ;  キャラクターワークの初期化を記述
18:         ;
19:         LD      (IX+MOD),1      ; Next move mode
20:         RET
21: ;
22: ;       <<< TEST MAIN
23: ;
24: TEST_MODE_1:
25:         ;
26:         ;  キャラクターのメイン処理を記述
27:         ;.
28:
```

リスト3　SIN関数移動

```
 1: ;
 2: ;          SIN関数移動サンプルプログラム（一部）
 3: ;
 4:
 5: ;          始めに移動量を決めるサンプル（直進）
 6:
 7:            ;  初期化部分
 8:
 9:            LD       A,(PL_X)
10:            LD       H,A
11:            LD       A,(PL_Y)
12:            LD       L,A
13:            CALL     DIRS_             ; プレイヤーの方向サーチ
14:            LD       (IX+DIR),A        ; 方向セット
15:            LD       (IX+SPD),$C       ; 速度セット
16:            CALL     SINC_             ; 移動量計算
17:
18:            RET
19:
20:            ;  移動部分
21:
22:            CALL     MOVE_             ; あらかじめ計算した移動量で移動
23:
24:            CALL     OUTC_             ; 画面外チェック
25:            JR       C,EBA_CLR
26:
27:            RET
28:
29: ;          毎回計算するサンプル（回転）
30:
31:            ;  初期化部分
32:
33:            LD       (IX+DIR),48       ; 初期方向セット
34:            LD       (IX+SPD),12       ; 速度セット
35:
36:            RET
37:
38:            ;移動部分
39:
40:            LD       A,(IX+DIR)
41:            ADD      A,2               ; 時計まわり
42:            AND      $3F
43:            LD       (IX+DIR),A
44:            CALL     SINM_             ; SIN関数移動
45:
46:            RET
```

純に座標に1を足しているのと同じです。ということは，1キャラクタ動くまでに16段階の速度があることになります。テキストベースのS-OSではこれくらいあれば十分でしょう。

11．POW/HPT：ダメージカ，生命力

このワークはキャラクターどうしがヒットした場合に参照，操作されます。ヒットが確定したとき，互いのHPTから相手のPOWを引くことになり，各キャラクターのプログラム上からはHPTをチェックして，負の値になっていれば死亡ということになります。

もちろん判定は各自で記述することになるので，HPTが負の値になったときの意味は自由に設定してください。あらかじめ生命力をユーザーワークに設定して，HPTを0にしておく方法を使えば，弾が当たるたびにキャラクターを発光させることもできます。その際，ユーザーワークに設定した生命力を減らすのを忘れないでください。

実際の使い方は，リスト4に示しておきます。

12．SXP/SYP/EXP/EYP：サイズ

ヒットチェックする際にヒット範囲が必要になりますが，このワークでそのサイズを指定します。はじめの2バイトで左上の，次の2バイトで右下の点を指定するわけです。基準になる点はキャラクターの絵の左上の点で，そこからの相対距離で指定してください。

このサイズに関しては，試行錯誤で決定していくのが普通でしょう。これといった基準はありません。

13．ユーザーワーク

ユーザーが自由に使えるワークです。8バイト用意してあるので，自由に使ってください。

これで，キャラクターワークの説明をひととおり行ったわけですが，理解してもらえたでしょうか？　6月号のサンプルプログラムと今回のサンプルプログラムを参照しながら読んでもらえれば，これといって難しいことはないと思います。要は「いかに使うか」という点が重要なのです。その点をこれからマスターしてください。

各種テーブルの構造

YGCSを使ううえで重要なデータがもうひとつあります。それはシステムが参照する各種テーブルです。これがないとYGCSはまったく意味をなさなくなるので，ある意味ではシステムの核といっても過言ではありません。

それでは順番にデータ構造について解説していきましょう。

1．キーテーブル

はじめは簡単なところで，マンインタフェイスである入力キーの定義テーブルです。リスト5を見てください。テーブル構造と実際のテーブルデータの例を挙げてみました。見てのとおり非常に単純な構造です。このテーブルについては悩むことはないでしょう。

2．キャラクターテーブル

前項で説明した，キャラクターIDで参照されるテーブルがこれです。ここには各キャラクターのプログラムの実行アドレスを登録します。それではリスト6を見てください。大まかに見て，

0〜3：プレイヤー関係
4〜B：敵本体関係
C〜F：敵弾関係
10〜　：そのほか

のように分類されます。途中ダミーでテーブルを確保しているのは，あとからプログラムを追加したとき，IDをずらさないでい

リスト4　衝突処理

```
 1: ;
 2: ;        衝突処理サンプルプログラム（一部）
 3: ;
 4:
 5: ;        単純な処理例
 6:
 7:          ; 初期化部分
 8:
 9:          LD      (IX+POW),1        ; ダメージカセット
10:          LD      (IX+HPT),0        ; 生命力セット
11:
12:          ; チェック部分
13:
14:          LD      A,(IX+HPT)
15:          OR      A
16:          JP      M,PB_CLR          ; 生命力マイナスなら消去
17:
18: ;        複雑な処理例
19:
20:          ; 初期化部分
21:
22:          LD      (IX+POW),1        ; ダメージカセット
23:          LD      (IX+HPT),0        ; ダミーの生命力セット
24:          LD      (IX+24),8         ; 生命力セット
25:
26:          ; チェック部分
27:
28:          LD      A,(IX+HPT)
29:          OR      A
30:          JP      M,PL_HIT          ; ダミーの生命力マイナス？
31:
32:          ; ヒット処理部分
33:
34:          LD      HL,PAT_PLD        ; ダメージパターン表示
35:          LD      (IX+PAT),L
36:          LD      (IX+PAT+1),H
37:
38:          LD      (IX+HPT),0        ; ダミーの生命力リセット
39:
40:          LD      A,(IX+24)
41:          DEC     A                 ; 生命力減少
42:          LD      (IX+24),A
```

リスト5　キー定義テーブル

```
 1: ;
 2: ;        キー定義テーブルサンプル
 3: ;
 4:
 5: ;        方向キーデータ
 6: ;         0 : UP
 7: ;         1 : UP-RIGHT
 8: ;         2 : RIGHT
 9: ;         3 : DOWN-RIGHT
10: ;         4 : DOWN
11: ;         5 : DOWN-LEFT
12: ;         6 : LEFT
13: ;         7 : UP-LEFT
14: ;        トリガキーデータ
15: ;         8 : BUTTON A
16: ;         9 :    "   B
17: ;        10 :    "   C
18: ;        11 :    "   D
19:
20: KEY_TBL:
21:          DB      '89632147'        ; 方向キー
22:          DB      'ZXCV'            ; トリガー
23:
```

リスト6　キャラクターテーブル

```
 1: ;
 2: ;        キャラクターテーブルサンプル
 3: ;
 4:
 5: CHR_TBL:
 6:          ;       Label             ID   Name
 7:          DW      PLAYER            ;  0 : Player
 8:          DW      PL_BULLET         ;  1 : Player bullet
 9:          DW      0
10:          DW      0
11:          DW      ENEMY_A           ;  4 : Enemy A
12:          DW      ENEMY_B           ;  5 : Enemy B
13:          DW      ENEMY_C           ;  6 : Enemy C
14:          DW      ENEMY_D           ;  7 : Enemy D
15:          DW      0
16:          DW      0
17:          DW      0
18:          DW      0
19:          DW      EN_BULLET_A       ;  C : Enemy bullet A
20:          DW      EN_BULLET_B       ;  D : Enemy bullet B
21:          DW      EN_BULLET_C       ;  E : Enemy bullet C
22:          DW      0
23:          DW      EXPLOSION_A       ; 10 : Explosion A
24:          DW      EXPLOSION_B       ; 11 : Explosion A
25:
```

いようにするためです。本来はもっと空けておくのが望ましいのですが，今回はサンプルということでこうなりました。

このテーブルでID 2番などをキャラクターワークで指定した場合，登録されているアドレスが0なので，当然のごとく暴走します。よってID番号指定は注意して行ってください。

3．キャラクター出現スケジュールテーブル

シューティングに限らず，どんなゲームでも敵キャラクターを出現させるにはスケジュールが必要です。そのスケジュールを記述しておくのがこのテーブルです。

リスト7を見てもらえばわかりますが，このテーブルは上記の2つのテーブルよりも構造が複雑になっています。まずは下記の基本構造を見てください。

DW　a,b+c＊$100,d,e+f＊$100

a＝出現カウント
b＝キャラクターＩＤ
c＝ワークサーチ数
d＝ワークサーチオフセット
e＝出現X座標
f＝出現Y座標

フォーマット自体は変更しないでください。

では各項目について解説します。

a．出現カウント

ここでいうカウントとは，システムが自動的にカウントアップしているカウンタです。システムがホットスタート（HOT）された直後にリセットされて，ジョブコントローラ（MAIN）が呼び出されるたびにカウントアップされていくわけです。

次に，そのカウンタの値がいくつになったらキャラクターを出現させるかを，記述します。0を書いておけば，スタート直後にキャラクターが出現するわけです。

b．キャラクターＩＤ

これは，キャラクターワークで説明したキャラクターIDの値を，そのまま書きます。要は出現させたいキャラクターのID番号を書けばＯＫです。

c．ワークサーチ数

キャラクターワークの空きを，いくつのワークから探すかを指定します。もしここに1が書いてあれば，1つのワークしかサーチせず，そこが使用中であればこのキャラクターは発生できなくなります。次に説明するオフセットと併せて，キャラクターワークを計画的に管理してください。

d．ワークサーチオフセット

キャラクターワークの空きを常にトップから探すのは，効率がよくありません。

そこで，キャラクターワークを論理的にいくつかに分けて管理することを考えます。その際，論理的に分けた各ワークのトップアドレスを指定するのがここです。

ワークサーチ数とワークサーチオフセットに関しては，リファレンスで詳しく解説しているので必ず参照しておくといいでしょう。

e．出現座標

これはもう解説する必要はありませんよね。出現させる座標をそのまま記述してください。

以上，とりあえずシステムの初期化に必要なテーブルを説明しました。これらの構造がわからないとYGCSを動かすことができません。サンプルプログラム（6月号）も参照しながら，構造を確認しておいてください。

表示データ構造

最初のほうで述べたように，キャラクターの表示用データ構造は大まかに3種類あります。

1つ目はサイズが1キャラクタのみの場合で，PATに表示データアドレスの代わりに直接キャラクタコードを書く方法です。この方法は弾などに使うといいでしょう。

2つ目はサイズが2×2と3×3の場合で，リスト8のはじめのデータのようにサイズ固定データです。

3つ目は上記以外のサイズのキャラクターを表示する場合で，2つ目の固定サイズデータの頭に，サイズ指定を付けたデータ構造です。どんなサイズでも扱えます。

以上3種類の表示データ構造をうまく使ってください。なお，このように場合分けしたのは，処理速度軽減のためです。つまり，サイズを自由に設定できる汎用ルーチンを用意するよりも，よく使われるだろうと思われるサイズのみを描画するルーチンを用意することで，キャラクター表示を高速化できるのです。

＊　＊　＊

さて，ワークとデータの構造に関しての説明はここまでとします。これ以外のデータやワークについては，おいおい説明していくつもりです。

次回は各キャラクターのプログラム方法を解説する予定です。

リスト7　出現スケジュールテーブル

```
 1: ;
 2: ;              出現スケジュールテーブルサンプル
 3: ;
 4:
 5: ; ワークサーチ・オフセット=スタートポイント×32
 6:
 7: CHR_GEN:
 8:         ;       COUNT:COD:SRCHCNT:OFADR:XP:YP
 9:         DW      $0000,$00+01*$100,$0000,17+25*$100      ; Player set
10:
11:         DW      $0020,$02+08*$100,$0120,08+04*$100      ; Enemy A set
12:         DW      $0030,$02+08*$100,$0120,12+04*$100
13:
14:         ; 以下続く
15:
```

リスト8　表示用データ構造

```
 1: ;
 2: ;              表示用データ構造サンプル
 3: ;
 4:
 5:         ; データのみ
 6: PL_DAT:
 7:         DB      ' A '     ; 3×3の例
 8:         DB      '<M>'
 9:         DB      ' Y '
10:
11:         ; サイズ指定データ
12: BS_DAT:
13:         DB      5,4       ; 5×4の例
14:         DB      '  I  '
15:         DB      ' (V) '
16:         DB      '<WnW>'
17:         DB      ' o¥o '
18:
```

リスト9　YGCS ver.0.30ヘッダファイル

```
 1: ; ==============================
 2: ;  Yu-ri Game Core System v0.30
 3: ;   for S-OS "SWORD" shooting
 4: ;     programmed by Yu-Ri.
 5: ; ------------------------------
 6: ;        SYSTEM EQU FILE
 7: ; ==============================
 8:
 9: ; ==============================
10: ;      SYSTEM CALL TABLE
11: ; ==============================
12: ; ***** System call
13: COLD_   EQU     $3000           ; System cold init (all init)
14: HOT_    EQU     $3003           ; System hot init
15: MAIN_   EQU     $3006           ; System job control
16: INIT_   EQU     $3009           ; System init
```

```
 17: RETN_    EQU     $300C           ; System return
 18: STAT_    EQU     $300F ; No use  ; System status read
 19:
 20: GENF_    EQU     $3012           ; Character generate table address set
 21: CHRF_    EQU     $3015           ; Character table address set
 22: BGDF_    EQU     $3018 ; No use  ; BG data address set
 23:
 24: VADR_    EQU     $301B           ; VRAM address get
 25: CADR_    EQU     $301E           ; CHR work address get
 26:
 27: PUSC_    EQU     $3021           ; Pause check
 28:
 29: RNDG_    EQU     $3024           ; Random value generate
 30:
 31: KEYF_    EQU     $3027           ; Key list address set
 32: KEYR_    EQU     $302A   ; SYS   ; Key read
 33: KEYE_    EQU     $302D   ; SYS   ; Key on edge check
 34: KEYG_    EQU     $3030           ; Key data get
 35:
 36: ; ***** Service call : VRAM(BG)
 37: VRAMT_   EQU     $3040   ; SYS   ; Vram display (all pages)
 38: VRAMC_   EQU     $3043           ; Vram clear (page:0-3)
 39: VRAMF_   EQU     $3046           ; Vram fill (page:0-3)
 40: VRAMS_   EQU     $3049           ; Vram scroll (page:3)
 41: VRAMA_   EQU     $304C   ; SYS   ; Vram address calc
 42:
 43: ; ***** Service call : OBJECT
 44: OBJI_    EQU     $3060   ; SYS   ; Object work init (all)
 45: OBJD_    EQU     $3063   ; SYS   ; Object display (all)
 46: OBJC_    EQU     $3066   ; SYS   ; Object clear (all)
 47: OBJS_    EQU     $3069           ; Object set
 48: OBJF_    EQU     $306C           ; Object free count
 49: STRF_    EQU     $306F           ; Work structure address set
 50: TNYS_    EQU     $3072           ; Tiny object set
 51:
 52: ; ***** Service call : CHARACTER
 53: CHRG_    EQU     $3080   ; SYS   ; Character generate
 54: CHRC_    EQU     $3083   ; SYS   ; Character control
 55: HITI_    EQU     $3086   ; SYS   ; Hit check init
 56: HITC_    EQU     $3089   ; SYS   ; Hit check
 57: MODC_    EQU     $308C           ; Mode control
 58: SINC_    EQU     $308F           ; Sin calc
 59: MOVE_    EQU     $3092           ; Move difference
 60: SINM_    EQU     $3095           ; Sin move
 61: DIRS_    EQU     $3098           ; Direction search
 62: OUTC_    EQU     $309B           ; Screen out check
 63: OPTM_    EQU     $309E           ; Option move
 64: ASI_     EQU     $30A1           ; Acceleration/slow down init
 65: ASM_     EQU     $30A4           ; Acceleration/slow down move
 66: DIRT_    EQU     $30A7           ; Direction trace
 67:
 68: ; ***** Service call : GAME
 69: SCOI_    EQU     $30C0           ; Score init
 70: SCOS_    EQU     $30C3           ; Score set
 71: SCOD_    EQU     $30C6           ; Score display
 72:
 73: ; ***** System work address & etc
 74: _VER_    EQU     $30E0           ; System version
 75: _WTOP_   EQU     $30E2           ; Work top
 76: _WBTM_   EQU     $30E4           ; Work bottom
 77: _CWRK_   EQU     $30E6           ; Character work top
 78: _VRM0_   EQU     $30E8           ; Vram 0 top
 79: _VRM1_   EQU     $30EA           ; Vram 1 top
 80: _VRM2_   EQU     $30EC           ; Vram 2 top
 81: _VRM3_   EQU     $30EE           ; Vram 3 top
 82: _HITW_   EQU     $30F0           ; Hit check work top
 83: _SCOW_   EQU     $30F2           ; Score work top
 84:
 85: ; ==============================
 86: ;     CHARACTER WORK OFFSET
 87: ; ------------------------------
 88: ; 1chr work size 32byts
 89: ;
 90: ; OFS SIZ  NAME  NOTE
 91: ;   0(1) : FLG : 00 - no use
 92: ;                01 - 1x1 chr
 93: ;                02 - 2x2  "
 94: ;                03 - 3x3  "
 95: ;                04 - nxn  "
 96: ;   1(1) : ATR : 0 - player
 97: ;                1 - option
 98: ;                2 - pl bullet
 99: ;                3 -
100: ;                4 -
101: ;                5 -
102: ;                6 -
103: ;                7 - 1/enemy
104: ;   2(1) : COD : CHR number
105: ;   3(1) : CNT : counter
106: ;   4(2) : XPS : X position
107: ;   6(2) : YPS : Y position
108: ;   8(2) : XDF : X difference
109: ;  10(2) : YDF : Y difference
110: ;  12(2) : PAT : pattern address
111: ;               (FLG = 01
112: ;                    CHR code)
113: ;  14(1) : MOD : move mode
114: ;  15(1) : *** :
115: ;  16(1) : DIR : direction
116: ;  17(1) : SPD : speed
117: ;  18(1) : POW : power
118: ;  19(1) : HPT : hit point
119: ;  20(1) : SXP : Start X pos
120: ;  21(1) : SYP :   "   Y pos
121: ;  22(1) : EXP : End X pos
122: ;  23(1) : EYP :   "  Y pos
123: ;
124: ;  24-31 : User work (free use)
125: ;
126: ; ==============================
127:
128: FLG      EQU     00
129: ATR      EQU     01
130: COD      EQU     02
131: CNT      EQU     03
132: XPS      EQU     04
133: YPS      EQU     06
134: XDF      EQU     08
135: YDF      EQU     10
136: PAT      EQU     12
137: MOD      EQU     14
138: DIR      EQU     16
139: SPD      EQU     17
140: POW      EQU     18
141: HPT      EQU     19
142: SXP      EQU     20
143: SYP      EQU     21
144: EXP      EQU     22
145: EYP      EQU     23
146:
147: ; ==============================
148: ;     CHR GENE TABLE FORMAT
149: ; ------------------------------
150: ;  0 (2) : CNT : Generate count
151: ;  2 (1) : COD : CHR number
152: ;  3 (1) : SCT : Search count
153: ;  4 (2) : OAD : Offset address
154: ;  6 (1) : XPS : X position
155: ;  7 (1) : YPS : Y position
156: ; ==============================
157:
158: ; ==============================
159: ;     KEY (EDGE) DATA FORMAT
160: ; ------------------------------
161: ;             bit 0 - Up
162: ;                 1 - Down
163: ;                 2 - Left
164: ;                 3 - Right
165: ;                 4 - Button A
166: ;                 5 -   "    B
167: ;                 6 -   "    C
168: ;                 7 -   "    D
169: ; ==============================
170: _KU      EQU     $01
171: _KUR     EQU     $09
172: _KR      EQU     $08
173: _KDR     EQU     $0A
174: _KD      EQU     $02
175: _KDL     EQU     $06
176: _KL      EQU     $04
177: _KUL     EQU     $05
178: _BA      EQU     $10
179: _BB      EQU     $20
180: _BC      EQU     $40
181: _BD      EQU     $80
182:
183: ; ==============================
```

▶ 全機種共通システムインデックス ◀

＊以下のアプリケーションは，基本システムであるS-OS“MACE”またはS-OS“SWORD”がないと動作しませんので ご注意ください。

▶シミュレーションゲームや(アクション)ロールプレイングゲームがなくなっていくような気がして寂しい。
五十嵐　正治(21)神奈川県

第22部 magiFORTH TRACER
第23部 ディスクダンプ＆エディタ
第24部 "SWORD" 2000 QD
連載 対話で学ぶmagiFORTH
特別付録 PC-8801版S-OS "SWORD"
■86年7月号
第25部 FM音源ミュージックシステム
付録 FM音源ボードの製作
連載 計算力アップのmagiFORTH
特別付録 SMC-777版S-OS "SWORD"
■86年8月号
第26部 対局五目並べ
第27部 MZ-2500版S-OS "SWORD"
■86年9月号
第28部 FuzzyBASIC発表
連載 明日に向かってmagiFORTH
■86年10月号
第29部 ちょっと便利な拡張プログラム
第30部 ディスクモニタDREAM
第31部 FuzzyBASIC料理法<1>
■86年11月号
第32部 パズルゲームHOTTAN
第33部 MAZE in MAZE
連載 FuzzyBASIC料理法<2>
■86年12月号
第34部 CASL & COMET
連載 FuzzyBASIC料理法<3>

1987

■87年1月号
第35部 マシン語入力ツールMACINTO-C
連載 FuzzyBASIC料理法<4>
■87年2月号
第36部 アドベンチャーゲームMARMALADE
第37部 テキアベ作成ツールCONTEX
■87年3月号
第38部 魔法使いはアニメがお好き
第39部 アニメーションツールMAGE
付録 "SWORD" 再掲載とMAGICの標準化
■87年4月号
第40部 INVADER GAME
第41部 TANGERINE
■87年5月号
第42部 S-OS "SWORD" 変身セット
第43部 MZ-700用 "SWORD" をQD対応に
■87年6月号
インタラプト コンパイラ物語
第44部 FuzzyBASICコンパイラ
第45部 エディタアセンブラZEDA-3
■87年7月号
第46部 STORY MASTER
■87年8月号
第47部 パズルゲーム碁石拾い
第48部 漢字出力パッケージJACKWRITE
特別付録 FM-7/77版S-OS "SWORD"
■87年9月号
第49部 リロケータブル逆アセンブラInside-R
特別付録 PC-8001/8801版S-OS "SWORD"
■87年10月号
第50部 tiny CORE WARS
第51部 FuzzyBASICコンパイラの拡張
第52部 X1turbo版S-OS "SWORD"
■87年11月号
序論 神話のなかのマイクロコンピュータ
付録 S-OSの仲間たち
第53部 もうひとつのFuzzyBASIC入門
第54部 ファイルアロケータ＆ローダ
インタラプト S-OSこちら集中治療室
第55部 BACK GAMMON
■87年12月号
第56部 タートルグラフィックパッケージTURTLE
第57部 X1turbo版 "SWORD" アフターケア ラインプリントルーチン
特別付録 PASOPIA7版S-OS "SWORD"

1988

■88年1月号
第58部 FuzzyBASICコンパイラ・奥村版
付録 石上版コンパイラ拡張部の修正
■88年2月号
第59部 シューティングゲームELFES
■88年3月号
第60部 構造型コンパイラ言語SLANG
■88年4月号
第61部 デバッギングツールTRADE
第62部 シミュレーションウォーゲームWALRUS
■88年5月号
第63部 シューティングゲームELFES II
第64部 地底最大の作戦
■88年6月号
第65部 構造化言語SLANG入門(1)
第66部 Lisp-85用NAMPAシミュレーション
■88年7月号
第67部 マルチウィンドウドライバMW-1
連載 構造化言語SLANG入門(2)
■88年8月号
第68部 マルチウィンドウエディタWINER
■88年9月号
第69部 超小型エディタTED-750
第70部 アフターケアWINERの拡張
■88年10月号
第71部 SLANG用ファイル入出力ライブラリ
第72部 シューティングゲームMANKAI
■88年11月号
第73部 シューティングゲームELFES IV
■88年12月号
第74部 ソースジェネレータSOURCERY

1989

■89年1月号
第75部 パズルゲームLAST ONE
第76部 ブロックゲームFLICK
■89年2月号
第77部 高速エディタアセンブラREDA
特別付録 X1版S-OS "SWORD"<再掲載>
■89年3月号
第78部 Z80用浮動小数点演算パッケージSOROBAN
■89年4月号
第79部 SLANG用実数演算ライブラリ
■89年5月号
第80部 ソースジェネレータRING
■89年6月号
第81部 超小型コンパイラTTC
■89年7月号
第82部 TTC用パズルゲームTICBAN
■89年8月号
第83部 CP/M用ファイルコンバータ
■89年9月号
第84部 生物進化シミュレーションBUGS
■89年10月号
第85部 小型インタプリタ言語TTI
■89年11月号
第86部 TTI用パズルゲームPUSH BON!
■89年12月号
第87部 SLANG用リダイレクションライブラリDIO.LIB

1990

■90年1月号
第88部 SLANG用ゲームWORM KUN
特別付録 再掲載SLANGコンパイラ
■90年2月号
第89部 超小型コンパイラTTC++
■90年3月号
第90部 超多機能アセンブラOHM-Z80
■90年4月号
第91部 ファジィコンピュータシミュレーションI-MY
■90年5月号
第92部 インタプリタ言語STACK
■90年6月号
第93部 リロケータブルフォーマットの取り決め
第94部 STACK用ゲームSQUASH!
第95部 X68000対応S-OS "SWORD"
特別付録 PC-286対応S-OS "SWORD"
■90年7月号
第96部 リロケータブルアセンブラWZD
■90年8月号
第97部 リンカWLK
■90年9月号
第98部 BILLIARDS
■90年10月号
第99部 ライブラリアンWLB
■90年11月号
第100部 タブコード対応エディタEDC-T
■90年12月号
第101部 STACKコンパイラ

1991

■91年1月号
第102部 ブロックアクションゲームCOLUMNS
■91年2月号
第103部 ダイスゲームKISMET
■91年3月号
第104部 アクションゲームMUD BALLIN'
■91年4月号
第105部 SLANG用カードゲームDOBON
■91年5月号
第106部 実数型コンパイラ言語REAL
■91年6月号
第107部 Small-C処理系の移植
■91年7月号
第108部 REALソースリスト編
■91年8月号
第109部 Small-Cライブラリの移植
■91年9月号
第110部 SLANG用NEWファイル出力ライブラリ
■91年10月号
第111部 Small-C活用講座（初級編）
■91年11月号
第112部 Small-C活用講座（応用編）
第113部 MORTAL
■91年12月号
第114部 Small-C SLANGコンパチ関数

1992

■92年1月号
第115部 LINER
■92年2月号
第116部 シミュレーションゲームPOLANYI
■92年3月号
第117部 カードゲームKLONDIKE
■92年4月号
第118部 オプティマイザO80実践Small-C講座(1)
■92年5月号
第119部 COMMAND.OBJ実践Small-C講座(2)
■92年6月号
第120部 COMMAND.OBJ2実践Small-C講座(3)
■92年7月号
第121部 関数リファレンス実践Small-C講座(4)
■92年8月号
第122部 ワイルドカード実践Small-C講座(5)
第123部 グラフィックライブラリ GRAPH.LIB
■92年9月号
第124部 O-EDIT&MODCNV
■92年10月号
第125部 SLENDER HUL実践Small-C講座(6)
■92年11月号
第126部 EDIT実践Small-C講座(7)
■92年12月号
第127部 MAKE実践Small-C講座(8)

1993

■93年1月号
第128部 EDC-Tの拡張
■93年2月号
第129部 BLACK JACK
■93年3月号
第130部 シューティングゲームコアシステム作成法(1)
■93年4月号
第131部 シューティングゲームコアシステム作成法(2)
■93年5月号
第132部 シューティングゲームコアシステム作成法(3)
■93年6月号
第133部 REVERSI
■93年7月号
特別付録 MSX用S-OS "SWORD"
■93年8月号
第134部 MACINTO-C再掲載
■93年9月号
第135部 7並べ
特別付録 SLANG再々掲載
■93年10月号
第136部 シューティングゲームコアシステム作成法(4)
■93年11月号
第137部 S-OSで学ぶZ80マシン語講座(1)
■93年12月号
第138部 エディタアセンブラREDA再掲載

▶高校の頃は部屋にX68000がある夢をよく見た。起きるといつも消えていた。
松尾 繁(20)福岡県

CGA入門キット「GENIE」(その2)

プロジェクトチームDōGA
かまた ゆたか

先月に引き続き，7月号の付録ディスク「GENIE」の「動きのデザイン」の使い方を解説します。動きのデザインは奥が深いので，そのコツなども紹介します。ついでに，CGAシステムのマニュアルの増刷のお知らせもあるから，見落とさないようにね。

はじめに

先月号の付録ディスクとして発表しましたCGA入門キット「GENIE」はいかがでしたか？ 全国のX68000のハードディスクを画像データでいっぱいにしているでしょうか？ 宇宙戦艦や宇宙戦闘機を大量にデザインした方も多いのではないでしょうか。

メカデザインにも飽きてきた方は，今月から，そのメカをガンガン動かしましょう。先月の基礎編では，初心者向きに懇切丁寧に解説しましたが，ひと月も使っていればもう慣れてきていると思いますので，今回はあまりくどい説明は省略します。

動きをデザインする

「GENIE」が叶えてくれる3つの願いのうち，「メカをデザインする」と「アニメーションを作る」の2つは，前回ひととおり解説しましたので，今回は「動きをデザインする」から始めましょう。

1）動きのデザインとは

以前からCGAシステムを使っていた方はともかく，この「GENIE」で初めて本格的にCGA制作を体験された方にとっては，「動きをデザインする」ということ自体が理解しにくいかもしれません。

動きとは，時間の変化に従って位置が変化することです。CGAの場合は，時間はフレーム数(1秒は20フレーム)で，位置は座標(X，Y，Z)で表現できます。よって，動きは，

1フレーム目に(0, 0, 0)
2フレーム目は(10, 0, 0)
3フレーム目は(20, 0, 0)
⋮
⋮
10フレーム目は(100, 0, 0)

のように記述できます。この例では，原点からX軸上を＋方向に動いていますね。しかし，このように1つひとつ設定するのは面倒なので，CGAシステムでは，

1フレーム目に(0, 0, 0)で，
10フレーム目は(100, 0, 0)で，
その間は，なめらかに移動する

というように，設定を省略することができます。

曲線上を移動する場合には，

1フレーム目に(0, 0, 0)で，
10フレーム目は(100, 0, 10)で，
20フレーム目は(200, 0, 100)で，
その間は，なめらかに移動する

というように，3点以上の位置を指定するだけで，その間はプログラムが自動的に適当な値にしてくれます。なお，プログラムが自動的に補間したフレーム以外の，人間が位置を指定したフレームをキーフレームといいます。この例でいえば，1，10，20フレームがキーフレームとなります。

「GENIE」の動きのデザインも，キーフレームを設定して，位置を変更する，という作業を繰り返すことで動きを表現します。もちろん，フレームごとに位置だけではなく物体の向きも同時に変更しますし，場合によっては大きさも変えることがあります。さらに，物体が動くだけではなく，カメラ(視点)や注目点が動くことで，よりダイナミックな映像を作ることができます。

2）起動

「GENIE」を起動して，メインメニューから，「動きをデザインする」に入ります。

写真1のメニューが出ますので，いちばん上の「新たに動きをデザインする」を選択してく

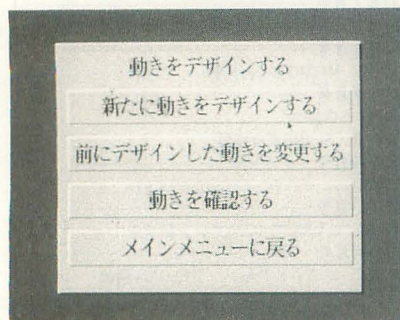



写真1 「動きをデザインする」のメニュー

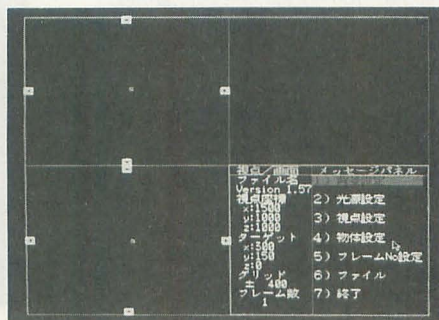



写真2 FFE.Xの画面

ださい。

すると「メカをデザインする」のところで見たFFE.Xと同じ画面になります(写真2)。しかし，よく見ると，メッセージパネルのなかに「2)光源設定」「5)フレームNo.設定」などというメニューが増えています。さらに，グリッドの値が「±400」となっています(「メカをデザインする」では100)。これは，平面図，側面図の青い1マスが400である，つまり平面図，側面図の表示範囲が縦横4倍に広くなったことを意味します。

写真3　動きのデザインではシンプル形状が表示される

写真4　1フレーム目の位置を設定する

では，先月号で制作した「OHX01」号を使って簡単な動きをデザインしてみましょう。遠くのほうから手前に向かってまっすぐ飛んでくるというものです。これなら単純な動きなので，1フレーム目と10フレーム目を設定するだけです。

3) 1フレーム目の設定

まず，動かす物体を設定します。「4)物体設定」に入り，「1)追加」で「OHX01.suf」を選択します。すると，平面図と側面図の中心に小さく「OHX01」が黄色で表示されます。

ここで，メッセージパネルの「作画」をクリックしてみてください。完成予想図に「OHX01」が表示されます(写真3)。でも，この形は，デザインしたものとは少し違っていますよね。機首はこんなに尖ってはいなかったはずです。これが「GENIE」の自動シンプル機能です。

形状デザインの場合は，かなり細かい部分まで表示する必要がありますが，動きのデザインのときは，大体どんなふうに見えるかがわかれば十分です。それよりも，位置を変更するときなどは，さっと消して，さっと描いてくれるほうが便利です。ですから，動きのデザインを行うときは，形状デザインしたデータとは別に，面数が少なく，おおざっぱな形がわかる程度の形状(シンプル形状)を使用します。

「GENIE」では，このシンプル版の形状を自動的に作成し，動きをデザインするときは，シンプル版のほうを表示するようになっているのです。CPUパワーが大きくないマシンをお持ちの方には，特にありがたい機能ではないでしょうか。

さて，現在のフレームナンバーは，左側のメニューのいちばん下の「フレーム数」の下に表示されています。ここでは「1」となっていますね。1フレーム目の「OHX01」はこの位置(原点)には決定しないでください。最初の位置としては，だいたい(−6000，−1000，−2000)ぐらいがいいでしょう。厳密にこの値にする必要はありません。「作画」してみると写真4のようになります。これで「決定」です。

4) フレームナンバーの変更

「4)終了」でFFEのルートメニューに戻り，「5)フレームNo設定」に入って，フレームナンバーを変更しましょう。

「メッセージパネル」には，

▲
ナンバー：　1
▼

決定
削除
中止

のように表示されています (写真5)。この状態でキーボードからフレームナンバーのところに「10」を入力し，「決定」を選択します。これで，10フレーム目の設定状態になりました。

ここで行うのは今回はこれだけですが，ついでにほかの機能もひととおり解説しておきましょう。

「▼」「▲」は，複数のキーフレームが設定されているとき，「▲」で前のキーフレームに，「▼」で次のキーフレームに移ります。あるキーフレームを変更したいが，何フレーム目だったかよく覚えていないときなど便利です。

「削除」は，キーフレームの削除を行います。たとえば1，10，20フレームをキーフレームとして設定していたのを1，15，20フレームに変更する場合は，まず10フレーム目を削除してから，15フレーム目を新たに設定します。

写真5　設定するキーフレームのナンバーを入力する

写真6　10フレーム目の位置を設定する

「中止」は，キーフレームの変更を行わず，そのままルートメニューに戻ります。

5）10フレーム目の設定

1フレーム目と同じ位置のままではアニメーションになりませんから，「OHX01」の位置を変更します。やり方は「メカをデザインする」のときの変更と同じなので，解説は簡単にしましょう。

「4)物体設定」の「2)変更」に入ります。いまの場合，変更の対象となる物体は1つしかありませんので，自動的に「OHX01」が選択されます。

「OHX01」が黄色く表示されますので，視点のすぐ下(2000，500，500)ぐらいに移動してください。「作画」すると，「OHX01」の後部がちらっと見えた状態になるはずです(写真6)。「決定」で終了します。

これで，向こうからこっちに向かってくる動きが設定できたわけです。

「4)終了」してメインメニューに戻ったら，「6)ファイル」「SAVE」「フレームソース」で，名前を入力してください。

確認する動きを選択してください
SAMP1 | SAMP2 | SAMP3 | SAMP4 | SAMP5 | SAMP6
SAMP7 | SAMP8 | SAMP9 | CUT01 | 中止

写真7　確認する動きのデータを選択する

「CUT01」とでもしましょう。名前はできるだけ5文字以内にするよう心がけてください。

最後に，「7)終了」でFFE.Xを終了します。

以上で，基本的な動きのデザインを理解してもらえたと思います。この例ではキーフレームを1フレームと10フレームの2つしか設定していませんが，同様に20フレームを設定すると，3点を通る曲線上を動いていくことになります。

6）動きの確認

動きのデザインをしたら，作画する前に「動きを確認する」を使いましょう。作画には時間がかかりますからね。

メニューのなかの「動きを確認する」をクリックすると，サンプルデータと共に先ほどセーブしたデータ「CUT01」のボタンがあるはずです。それを選択します(写真7)。

「OHX01」が向こうから飛んでくる様子が，次々にワイヤーフレームで作画されます(写真8)。10フレームの作画が終わると，アニメーションします。

この状態で，

「ESC」　：終了
「SPACE」　：一時停止，コマ送り
「RETURN」　：一時停止解除
「L」　：15kHz－31kHzの切り替え
「ROLL UP」「ROLL DOWN」
　：アニメーション速度の変更
「HOME」　：1フレーム目の状態で一時停止

といった機能が使えます。とりあえず動きが確認できたら，「ESC」で終了してください。

実際にオリジナルの動きをつける場合は，FFE.Xで動きをデザインして，ワイヤーフレームで確認して，気に入らないところを再び修正する，という作業を繰り返します。

勝手にゲームレビュー

この連載中でゲームレビューをするのは2回目です。前回は3Dワイヤーフレームの「STAR WARS」で，3Dの勉強になると思って掲載しました。今回はCGAには全然関係ありません。ただ単に，DōGAの一翼を担うKMC(京大マイコンクラブ)がゲームを出すということなので，ここで紹介するだけです。私(かまた)がレビューを書くと身びいきしそうなので，ハイスクール飯干君にレポートしてもらいました。

＊　＊　＊

はい，今回，パズル「ドリーミング」を紹介する飯干です。はじめ，KMCがゲームを作ったと聞いたときは，3Dのキャラクターを使ったシューティングゲームか，難しい技術を使ったゲーム(シャープのX68000芸術祭の「RUSH!」みたいな)なんかを想像したのですが，実にシンプルなパズルゲームとは意外でした。

このゲームは，ジグソーパズルの一種です。ただ，ピースは1個ずつばらばらではなく，数個(多いときは10個以上)がつながっています。プレイヤーは女の子(僕には忍者にしか見えないけどなぁ……)を操作し，そのピースの列の端をつかんで引きずって歩きます。ピースを持って，ぞろぞろ歩いている様子はドラクエのパーティのようです。女の子がグネグネ曲がれば，ピースたちも曲がってついてきます。そして，それらのピースすべてがぴったり絵と合わないといけないのですから，もう大変です。

このゲームは，全46面あります。最初はとっても簡単で，楽勝で進むのですが，面が進むにつれて「嫌らしさ」がだんだんエスカレートしていきます。ピースの数が増えたり，制限時間が短かったり，全ピースがくっついている面もあります。私は，草原のなかにウサギが1匹いる「Rabbit」(7面)や，そこらじゅう同じような窓だらけの「Babel」(14面)で，まずつまずきました。後半の「嫌らしさ」はすさまじく，45面など白と黒だけのフラクタル画像！　すぐイライラする人は，マシンが極悪に見えてくるでしょう。

でも，無限コンティニューなので，根性があればきっとクリアできます。また，もう一度起動すると，ピースのつながり方が変わるので，何度でも遊ぶことができます。

このように，ルールが単純なのでとっつきやすく，簡単にハマるゲームです。見た目は地味ですが，ゲームバランスもよく，意外と楽しめるでしょう。　（ハイスクール飯干）

「ドリーミング」1,500円(税込)
販売元：TAKERU
開発元：KMC

写真８　「OHX01」の動きがワイヤーフレームで作画される。２～９フレームは自動的に補間されたもの

写真９　アニメーションはこうなる

作画設定変更

動きを確認したら，「メインメニューに戻る」のあと，「アニメーションを作る」で，いまの「CUT01」を作画してみましょう。

「アニメーションを作る」「作画」「CUT01」で実行して，必要に応じて使用するメカを変更したり，色調を変更したりしてみてください。作画が終了したら，「アニメーション再生」で「CUT01」を選択します（写真9）。

このあたりの説明は先月号で行いましたので，今回は「作画設定変更」を解説しましょう。「作画設定変更」のメニューには，「アンチエイリアス」「スムースシェーディング」「誤差拡散ディザリング」「背景の星」という項目があります(写真10)。

表１　サンプルデータ「samp1」の作画時間（数値演算プロセッサ入りXVI(16MHz)による）

条件	作画時間(秒)	倍率
すべて「なし」	254	1
アンチエイリアスあり	558	2.2
スムースシェーディングあり	342	1.3
誤差拡散ディザリングあり	386	1.5
背景の星あり	274	1.1

「アンチエイリアス」というのは，低い解像度のときに斜めの線がギザギザになるのを，ぼやけさして目立たなくする処理です。これで，見た目の画質がかなり向上されます。

「スムースシェーディング」は，複数のポリゴンをなめらかに色を変えていくことで，曲面として表現する処理です。曲面の多いメカに効果があります。

「誤差拡散ディザリング」は，６万５千色しかないX68000で，擬似的に200万色出るようにする処理で，マッハバンドが出なくなります。しかし，できる画像データが非常に大きくなるのが欠点です。

「背景の星」は，文字どおり，背景の部分に宇宙空間のような星を表示する機能です。

すべて「あり」に設定したほうが仕上がりはよくなりますが，作画に時間がかかります。表１は，サンプルデータ「samp1」を，68881入りXVI(16MHz)で作画させたときの実測値です。画像によって，この値はかなり変動しますが，参考程度にご覧ください。

テストの作画の場合は全部

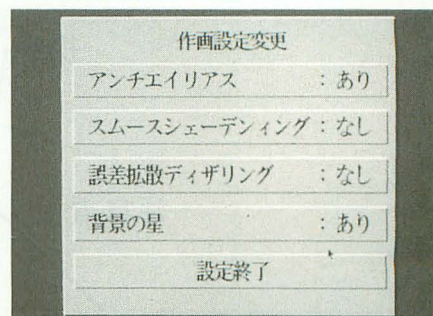



写真10　アニメーション「作画設定変更」メニュー

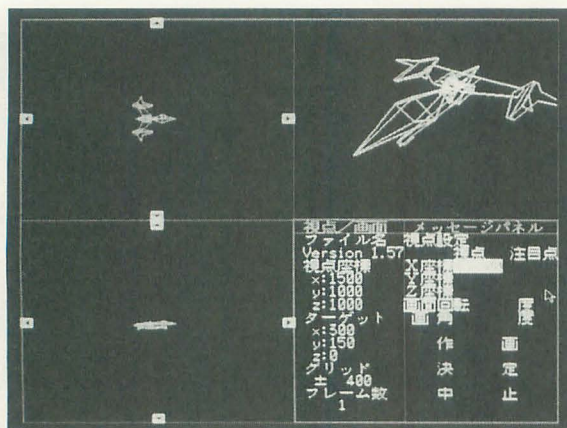



写真11　視点設定画面

「なし」にするとか，通常は「アンチエイリアス」と「背景の星」は「あり」にするとか，あるいは，CPUパワーに自信のある方や時間に余裕がある場合は全部「あり」にするといったように，状況に応じて使い分けるとよいでしょう。

視点の動き

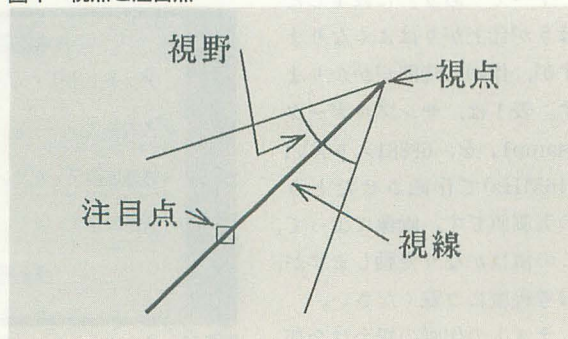
単に物体を動かすだけではなく，キーフレームごとの視点(カメラ)の位置や向きを変えてみると，視点が動き回るダイナミックな映像を作ることができます。

FFE.Xの平面図，側面図などで，図1のような3本の直線で表されるのが視点です。視線は赤で表示され，紫の2本の線で挟まれる領域が視野となります。視線の赤線上に，小さな赤い□マークがありますが，これは注目点です。見る方向を変更するには，視点を動かす方法と，注目点を動かす方法の2つがあります。また，両方を動かしてもかまいません。

視点の動かし方を練習する前に，とりあえず「OHX01」でもなんでもいいので，物体を1つ原点にでも置いてください。なぜなら，何か見るものがないと，見方を変更してもわかりませんから。

1）視点の変更

準備ができたら，FFEのメインメニューから，「3)視点設定」に入ります。

図1　視点と注目点

視点設定

	視点	注目点
X座標	1500	
Y座標		
Z座標		
画面回転	度	
画角	度	
作画		
決定		
中止		

と表示され，視点のX座標を示す「1500」が反転していると思います(写真11)。

この反転している(カーソルがある)ところが，「視点」の列である場合，視点の位置を変更する状態です。逆に，カーソルが注目点にある場合は，注目点を変更する状態です。

初期状態では現在の視点，注目点の座標などが表示されていませんが(考えてみると変な仕様ですね)，カーソルを動かせばちゃんと表示されます。カーソルを動かすのは，キーボードのカーソルキーでも，マウスで表示位置をクリックしてもかまいません。

それでは，カーソルを「視点」の列に戻して(X，Y，Z座標のどれでもよい)から，視点を動かしてみましょう。方法は，とっても簡単です。平面図，側面図などで，視点をもっていきたい位置をクリックするだけです。クリックすると，瞬時に視線と視野を示す赤と紫の線が移動します。前後左右に動かすときは，平面図，上下に動かすときは側面図で行うとよいでしょう。

視点を画面上の好きなところに動かしたら，「作画」をクリックしてどんな画像になるか見てください。気に入らなければ，また視点を変更しますが，その場合「作画」に移ってしまったカーソルを，「視点」の列に戻すのを忘れないでください。

物体の動かし方もそうでしたが，視点の位置の変更も，マウスだけでなく，キーボードで直接数値を与えてやることができます。変更したい値のところにカーソルを移動したのち，キーボードで数値を入力してリターンキーで確定します。

2）注目点の変更

注目点の変更も，視点の変更とまったく同じです。「注目点」の列のどこかにカーソルがある状態で，平面図などでクリックすると，その位置に注目点が移ります。

ただ，注目点を変な位置に設定すると，「作画」しても何も表示されません。当然ですね。ちゃんと，視野のなかに物体が入るように設定してください（写真12)。

3）画面回転，画角の変更

「視点設定」のなかには，「視点」「注目点」以外に，「画面回転」「画角」というパラメータがあります。しかし，これは初心者の方が無理して使う必要もないので，忘れ

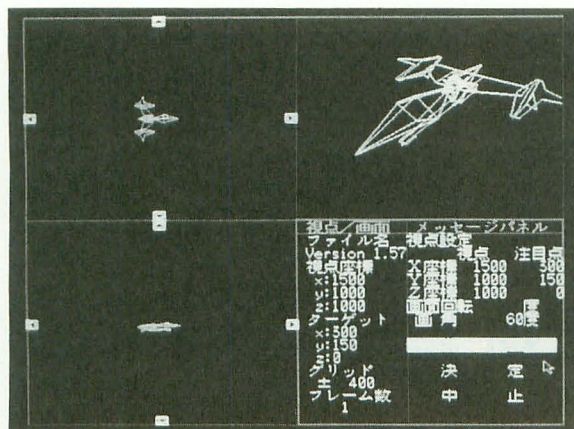



写真12 視点や注目点を変更してみる

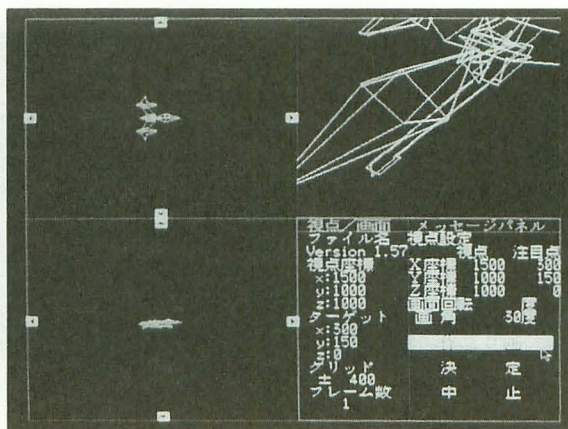



写真13 画角を変える(30度)

ていただいても結構です。

簡単に解説すると，「画角」とは，視野を示す紫の２本の線が交わる角度です。つまり，画角が大きいと広い範囲が見え，小さいと狭い範囲しか見えません（写真13)。しかし，単に広い範囲が見たければ，視点の位置を後ろに下げればすむことです。

一般に，画角が大きくなると，遠近感が強調され，巨大感がでます。「GENIE」では，画角は通常60度に設定されていますが，これは現実のカメラの画角などと比べると，かなり大きな値といえます。

次に「画面回転」ですが，この角度が０のときは，視点の上下方向と，空間のＺ軸方向が一致していますが，この角度が変化すると，首を傾けて見たような画面になります(図２)。戦艦に突入する戦闘機のコックピットからの画像を作る場合など，バンクを表現するには有効な手段です（写真14)。

しかし，バグに近い仕様というか，FFE.Xの画面回転の計算の仕方は，REND.Xなどの計算方法と異なるため，仕上がりの画像がかなり異なってしまいます。それぞれの計算アルゴリズムまでは解説しませんが，視線がＸ軸に近いときは，そんなに差はないので，この値を変更するときは，視線がＸ軸方向を向いているときだけにするのが無難でしょう。

なお，「画角」や「画面回転」の値は，マウスで指定することができません。カーソルを移動したあとは，キーボードで数値を入力してください。

ところで，視点や注目点などの変更は，「メカをデザインする」でも使えます。下や後ろから見たときのバランスをチェックするために，視点を変更しながらメカをデザインするとよいでしょう。

動きのデザインのコツ

1）キーフレームを少なくする

これは，非常に重要なポイントです。ずばりいうと，１カットのキーフレームは，基本的に３つだと思っていてよいでしょう。

２つでは，直線にしか動きません。３つあれば曲線になります。４つになるとＳ字のように左右１回ずつ曲がるという動きが表現できますが，そのようなカットは稀です。５つ以上は，まず無意味です。

とはいっても，初心者の方はつい，キーフレームを多くしてしまいます。たとえば，初め１，10，20フレームをキーフレームに設定して動かしてみたが，10～20フレームのあたりで思ったより右にきてしまうという場合，つい15フレーム目に新しいキーフレームを設定して，強

図2 画面回転(０度と30度)

画面回転30度

画面回転60度

画面回転90度

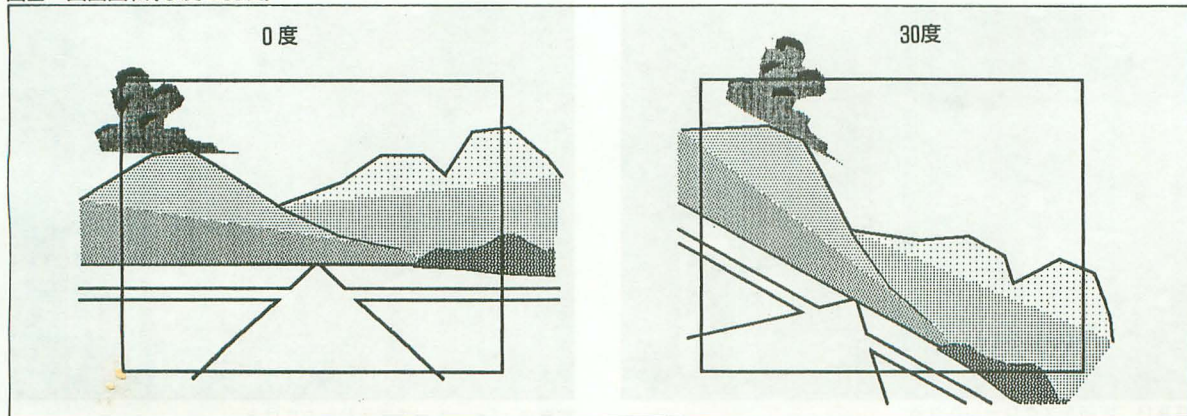
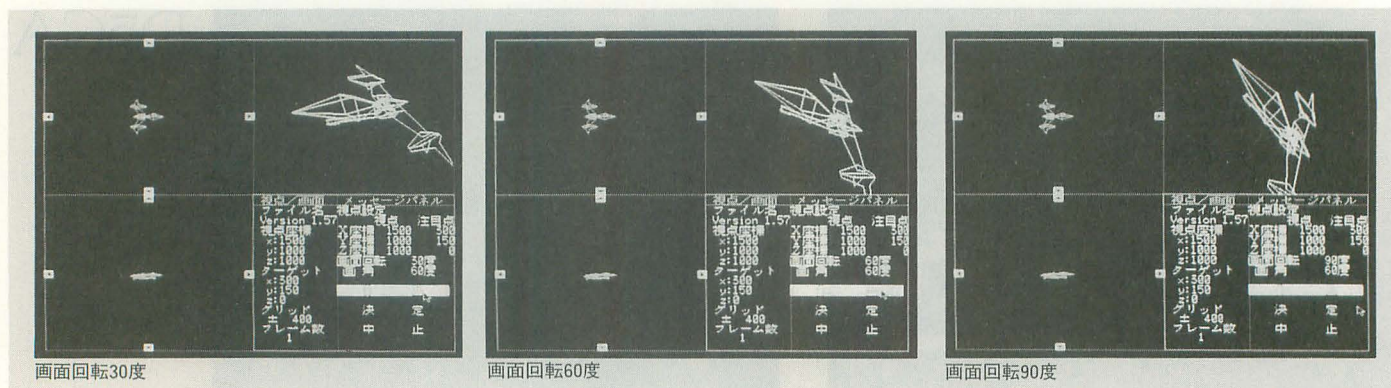



写真14　画面回転を変える

引に右へ行かないようにするのです。そうすると，確かに15フレーム目の前後では思ったとおりの位置にくるのですが，15～20フレームでこんどは左に行き過ぎる……そこでさらに18フレーム目をキーフレームにして……という感じで，どんどん悪化していきます。

このような場合は絶対にキーフレームを増やさないでください。キーフレームが多くなると，だんだんグニャグニャとした振れの大きな動きになり，制御できなくなります(この現象をスプラインのバタツキという)。

3つのキーフレームで思ったように動かなかったら，まず現在のキーフレームの位置をいろいろ変更して，それでだめならキーフレームのフレーム数を変更してみたり(例：1，7，20フレーム)するのが正しい方法です。

2）こまめにメモ

「メカをデザインする」のところでも述べましたが，FFE.Xには，ほかのフレームのデータを参照するという機能がほとんどありません。

たとえば，キーフレームが1，10，20フレームで，加速しながら飛んでいくカットを作る場合，1～10フレームの移動量より，10～20フレームの移動量のほうを大きくしなければなりません。ですから，20フレーム目の位置を設定するときには，10フレーム目の位置だけでなく，1フレーム目の位置も把握しておく必要があります。

ということで，物体の位置を変更して，「決定」をクリックする前に，その位置，向きなどはメモしておくようにしてください。

3）図面拡大

実際に自分で動きのデザインを始めるとすぐ気がつくと思いますが，戦闘機が派手に飛び回るカットを作る場合，平面図などが表示する領域は狭すぎます。しかし，これを解決する方法はいくつかあります。

まず，必要な方向にスクロールする方法があります。平面図などの上下左右にある「▲」のマークをクリックすると，そちら方向に画面がスクロールします（写真15）。ただし，このスクロールの方向は，なぜかCAD.Xとは逆になっていますので，CAD.Xに慣れている方には違和感があるでしょう。FFE.Xでは，クリックした方向に新たな余白が現れるので，たとえばX軸の＋側が見たければ，X軸の＋側の「▲」マークをクリックしてください。

次に，縮尺を変え，表示範囲を大きくする方法があります。この方法のほうが楽なのですが，表示される物体も小さくなって，判別がつきにくくなるという欠点もあります。キーボードの「＝」を押してください。表示範囲が1段階広くなって，1マスが1000になります（写真16）。逆に，細部を見たいときは，「＋」を押してください（写真17）。

まったく別の発想としては，戦闘機を小さくするという方法もあります。物体を置くときに，X，Y，Zの拡大の値をそれぞれ0.1ぐらいにするわけです。手間はちょ

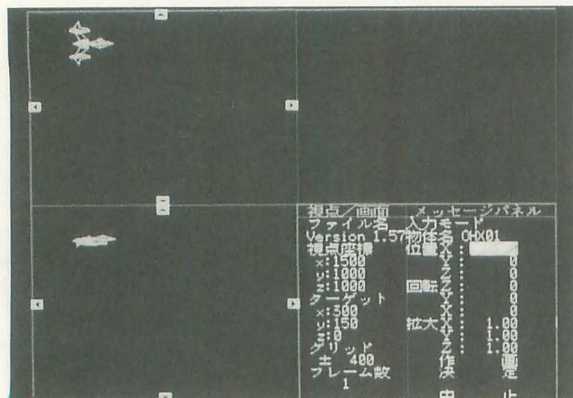



写真15　画面をスクロールさせる

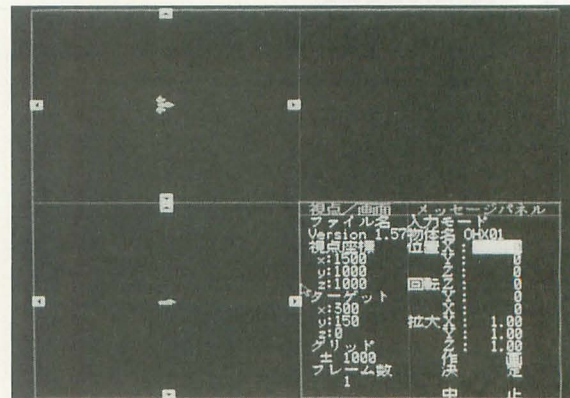



写真16　「＝」キーで表示範囲を広げる

っと増えますが，なかなか有効な手段です。

4）自然な向きの設定

大気中を飛ぶ飛行機と，宇宙空間を飛ぶ宇宙戦闘機の動きは，全然違うはずです。しかし，見た目に自然な感じにするためには，やはり飛行機と同じように動かさなければいけません。

まず，曲線上を飛ぶ場合，機首はちゃんとその曲線の接線方向を向いておく必要があります。ただ，どのような曲線にするかが決まったあとでないと，向きも決まりません。ですから，とりあえず向きは気にせずに，各キーフレームの位置を決定し，それから改めて最初のキーフレームから向きだけつけていくというのも，よくやる方法です。

向きを変えるというのは，「回転」の値を変化させることです(写真18)。ところで，位置がX，Y，Zの順番に対して，回転の順番がZ，Y，Xになっているのに気がついていましたか？　実は，位置はX，Y，Zのどの順番に与えても結果は同じですが，回転はやっかいなことに，順番が違えば最終的な向きが異なります。

それで，どのような順番で回転を与えるのがいちばん人間にとって理解しやすいかを検討した結果，このZ，Y，Xという順番になりました。こうすると，各値は，飛行機や船などの専門用語である，ピッチ，ロット，バンクとだいたい同じ考え方になるからです。

まず，平面図を見ながら，その物体が移動する曲線を頭の中で描いてみます。そして，その曲線に添うようにZの値を与えます。デフォルトは0度で，画面右(X軸の＋方向)を向いています。＋90で，画面上(Y軸の＋方向)を向きます。与える数値は－でも構いません。

「曲線がどんな感じになるかよくわからん」とダダをこねる人は，単純に，機首を次のフレームの位置のほうに向かせるだけでも結構です。その場合，できたアニメーションを見ると，車が後輪をズルズルすべらせながら曲がるような動きになりますが，宇宙空間なんて抵抗がないからすべって当たり前です。友達に「何か変だ」といわれたら「よりリアリティを追求するための演出だ」といい張ってください。

次に，側面図を見ながら，Yの値を決めます。側面図

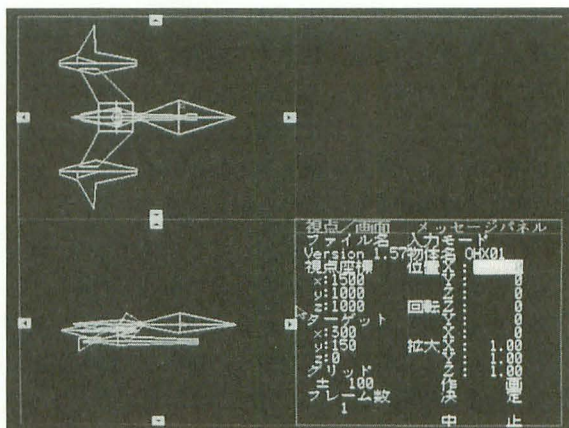



写真17　「＋」キーで細部を見られる

上でも曲線上を動くなら上記と同様に設定し，ほとんど直線上ならその直線の傾きに合わせて設定すればよいのです。

最後にXの値ですが，これはほかの2つとは考え方が少し異なります。まず，その物体のコックピットに乗ったところを頭に描き，右に旋回するときは＋，左に旋回するときは－の値を設定します。角度は最大90度で，ゆっくりとした旋回をするところでは小さい値，急な旋回をするところでは大きな値となります。もう少し厳密にいうと，急な旋回をする手前で大きな値となります。飛行機は，機体を傾けてから曲がり始めるからです。

5）光源の変更

そういえば，まだ解説していない機能がありました。FFE.Xのルートメニューのなかの「2)光源設定」，つまり，光の当たり方を設定する機能です。

CGAシステムでは，通常の平行光線のほか，点光源やスポット光源を複数設定できますが，FFE.Xでは，平行光線を1つしか置けません。平行光線は，カラーとベクトルが指定できます。ベクトルとは，光が射している向きで，カラーは光の色のことです。変更したい値のところにカーソルをもっていって，キーボードから数値を入力してください。

ベクトル(X，Y，Z)によって，方向を表現することは，数学の授業で習ったと思います。たとえば，(1，0，0)はX軸の＋方向を意味します。ここでは平行光線

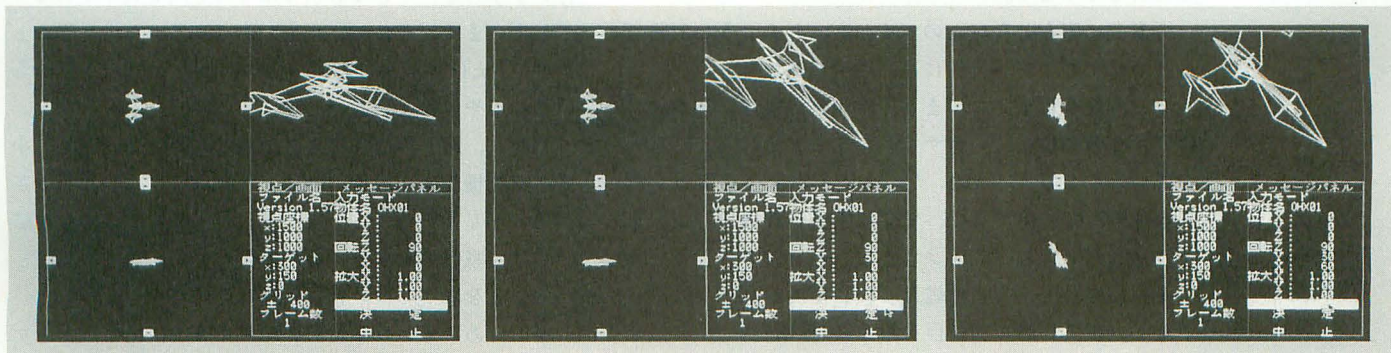



写真18　回転させてみる

のベクトルは，デフォルトでは(－3，－2，－4)に設定されています。つまり，物体をX軸の＋方向から見たとき，だいたい右斜め上から光が当たるような感じです。

視線と40度ぐらいずれて，右(あるいは左)斜め上から光が当たるというのが，最も基本的なライティングです。ある程度の陰影ができ，自然な感じになります。

これに対して，視線と光源のベクトルが完全に一致してしまうと，光が当たっていない面がまったく見えず，陰影の乏しい絵になってしまいます。また，視線と光源ベクトルが逆方向になっていると，物体の大部分が影になり，物体の陰影が乏しいだけでなく，見にくくなってしまいます。

もっとも，どのカットも光が右斜め上から当たっているというのも，現実にはありえないことです。演出のひとつとして逆光を用いる場合もあります。たとえば，巨大な戦艦が浮遊している感じを強調するために，逆光気味に，下から光を当てるというのはよくやるテクニックです。その例として，サンプルアニメーション「SAMP5」をご覧ください。

「SAMP5」では，光源に少し青い色をつけています。画面の下に青い惑星があるというイメージを出してみました。光源に色をつけると，画面全体に同系色の割合が増え，まとまりが出ます。あまり使う機会はないと思いますが，そのカットの状況を強調するには，たとえば緊迫感や冷たい感じを出すために青い光源を使うといった手法もあるでしょう。

なお，光源の設定は，キーフレームごとにはできません。1カットを通じて同じ向き，色になります。

戦艦の表現(形状)

宇宙戦闘機と宇宙戦艦の違いをどうやって表現するか考えてみましょう。方法は2種類あります。ひとつは，形状デザイン上の問題で，もうひとつは，動き方の問題です。

話は少し脇にそれますが，形状デザインも表現の一翼を担うものですから，ここで少し解説しましょう。まずは形状デザインにおける，宇宙戦艦と宇宙戦闘機の違いを注目してみます。

両者共に実在しないものですから，どのような形状でもよいはずですが，実際には，現実にある物体の形状に大きく影響されます。つまり，宇宙戦闘機は，基本的に実在する戦闘機っぽいデザインにするとそれらしく見えるということです。また，「STAR WARS」のような有名な映画などに出てきて，すでに宇宙戦闘機の形状として認知されているデザインでもよいわけです。

それに対して，宇宙戦艦はまだ自由度が高く，船だけでなく，現実にある何か巨大なもの(建築物など)に似せるという方法がとれます。もちろん映画やアニメで認知されている宇宙戦艦のデザインも有効です。

次に，翼とコックピットに注目する方法があります。機体全体に占める翼の割合が多いと戦闘機に見えます。また，コックピット部がはっきりとわかり，これが大きいとやはり戦闘機に見えます。逆に，艦橋部がはっきりとわかるデザインにすると戦艦らしくなります。そして，艦橋部が小さいほど，大きな戦艦になります。

同様に，砲台で区別する方法もあります。機体と同じぐらいの大きさの砲台が数個ついていれば戦闘機で，非常に小さい砲台が数多くついていると，戦艦っぽいわけです。でも，砲台を数多くつけるのは，面数が多くなりすぎますので，あまり現実的ではありません。

最後に，戦艦の巨大さを表現するのには，船体の一部だけディテールを細かくするという方法があります。「STAR WARS」のスターデストロイヤーやデススターなどは，細部の細部まで細かな凹凸や窓がついていて，それが全体の巨大感を出す効果がありました。しかし，あれをCGAで行おうとすると，非常に面数が増え，とても作画できません。だからといって，全体的にディテールをちょっとだけ細かくしても，面数の割に巨大感が出ません。そこで，船体の大部分は面数の少ないシンプルな形状にとどめておいて，艦橋付近やエンジン，また中心部近くの比較的目につきやすい一部分だけ，小さいパーツをたくさん並べてやるのです。

この手法は，ポリゴンのシューティングゲームなどにもよく使われています。一度は試す価値があるでしょう。

戦艦の表現(動き)

動きにおける戦艦と戦闘機の違いは，まず戦艦はすばやく動かないということです。特に，回転などはほとんど行いません。戦艦を見せるときは，視点や注目点もすばやく動かさないのが基本です。ただし，戦艦に突入する戦闘機のコックピットからの映像や，その戦闘機をフォローする映像など，視点や注目点の動きが戦闘機を意味する場合は例外です。

次に，大きさです。「GENIE」は，戦艦をデザインするときも，戦闘機をデザインするときも，同じスケールで行いますので，できる形状もほとんど同じ大きさになってしまいます。そこで，両方が出てくるカットを作るときは，例えば戦闘機は各軸方向に0.1倍，戦艦は数倍ぐらいにしてやればよいわけです。

しかし，遠くのほうに大きな戦艦があって，近くに小さな戦闘機がある場合，画面上では同じぐらいの大きさになってしまいます。つまり単に拡大率で大きくしただけでは，戦艦の大きさは表現できません。遠くにあるということを表現しなければいけないのです。

遠くにあるということを表現する方法はいくつかあります。まず，先ほどの話と重複しますが，ゆっくりと動

かすことは，遠くにあることを表現するひとつの手段です。目の前を通り過ぎる車は猛スピードに見えますが，遠くを飛んでいる飛行機は，ほとんど動いていないように見えます。画面上を２つの物体が横切る場合でも，１つはゆっくり，ほとんど動いていないぐらいで，もう１つが一瞬で横切ると，その大小に関わらず，前者が遠くにある戦艦，後者が手前の戦闘機になります。

もうひとつは，視点から遠くにある物体までの空間を，ほかのものが移動することで表現する方法です。手前から奥に向かって戦闘機が飛んで行って小さくなるその向こうに戦艦があれば，戦艦までが非常に遠いことがよく分かります。逆に，戦艦から戦闘機が手前に飛んでくるというパターンもあります。

また，視点の位置を大きく移動する方法もあります。注目点はほとんど移動しないか，視点と平行に動かします。こうすると，遠くのものはほとんど動かないのに対して，視点の近くのものは見え方が変化しますので，距離感が出ます。

おわりに

さて，この「GENIE」を使えば，宇宙バトルもののCGAが簡単に作れると思います。だからといって，当チームがこのジャンルのCGAを奨励しているわけではありません。単に，宇宙戦闘機や戦艦が飛び回るCGAが，最も初心者向きだと思っただけです。

むしろ，当方が主催するCGAコンテストでは，このようなジャンルは誰にでもできるといって，審査が厳しくなる可能性もあります。単に「GENIE」で作った数カットをつなぎ合わせてもそれだけでは入選は難しいでしょう。しかし，その作品に，なんらかの形でプラスαがあれば，「GENIE」を使ったかどうかに関わらず，それなりに評価されるでしょう。

結局，コンテストにおいて評価されるのは，何を使ったかではなく，各自の行うプラスαについてなのです。

さて，当初は「GENIE」の話は今回で終わりにするつもりでしたが，入門キットということで初心者向きに細かい部分まで説明しているうちに，ページ数がなくなってしまいました。

次回は，「GENIE」の応用編と４月号のアンケートの集計結果のほか，たまっているユーザーからのお手紙特集でもしたいと思います。

それでは，来月まで「GENIE」でハードディスクをいっぱいにしておいてくださいね。

CGAシステム マニュアル増刷決定！

「GENIE」によって，初めてCGAの楽しさを知った方も多いでしょう。しかし，「GENIE」だけでは作品は作れません。そうです，DōGA CGAシステムが必要です。しかし，CGAシステムのマニュアルは，すでになくなっている……。これでは，空きっ腹の人間に匂いだけかがせておいて，食べさせないのと同じだ！

ということで，マニュアルの増刷を，いまさら，この時期になって行うことになってしまいました。当然ですが，これが最後の増刷です。ver.３の開発が完成するまで(いつのことやら)，絶対に二度とありません。

マニュアルを手に入れる最後のチャンスです。まだ手に入れていない方は，この機会をお見逃しなく！　知人でマニュアルを欲しがっていた方がいたら，「Oh!Xで最後の募集をしていた」と教えてあげてください。

同時に，バージョンアップサービスも行うことになりました。２年前に付録ディスクとしてCGAシステムを発表してから，その後いくつか新しいプログラムも加わりましたし，各ツールも多々バージョンアップしています。CGAマガジンで一部のツールは発表しましたが，ユーザーによって持っているバージョンがまちまちというのも問題です。

そこで，最新のプログラムをまとめ，追加マニュアル(100ページぐらい？)をセットにして発表します。追加マニュアルのほうも，まとめて１回しか印刷しませんので，CGAシステムをお使いの方は，この機会にぜひ申し込んでください。

ところで，問題になってくるのが実費の計算です。マニュアルなどの印刷物は，何冊作るかで１冊あたりの単価は全然違ってきます。ですから，どれくらいの申し込みがあるかわからなければ，実費が予測できないのです。もう大部分の方は入手されているはずですから，少なくとも２年前の最初の申し込みのときよりは大幅に少ないでしょう。すると，かなり割高になることが予想されます。特に追加分のマニュアルは，版から起こさないとなりません。申し込みを先に行って，あとから値段を決めるというわけにもいかないし。う～む，困った。まぁ，採算が取れないほど申し込みが少なかったら，増刷を中止して返金するという手もあるし……(面倒だな)。えい，もう決めるしかない！　ということで，けっこうぎりぎりの線を出してみました。

[申し込み内容]

- マニュアルコース
  - 旧マニュアル(800ページ)
  - 追加マニュアル(約100ページ)
  - 最新プログラム
- バージョンアップコース
  - 追加マニュアル(約100ページ)
  - 最新プログラム

[実費]

マニュアルコース　　　　：5,000円
バージョンアップコース：2,500円

[申し込み期間]

1994年７月18日～８月31日

[発送期間]

９月中に印刷，発送は10月以降

[申し込み方法]

本誌綴じ込みの郵便振替用紙を使用

[注意事項]

・第６回のCGAコンテストビデオを申し込まれた方は，ビデオの封筒のラベルに書いてあったDōGA登録ナンバーを，住所・氏名欄に記入してください。

・払込人住所，氏名欄には，ご自分(発送先)の住所，電話番号を丁寧に明記してください。難しい漢字にはふりがなをふってください。

・裏面の「マニュアルコース」「バージョンアップコース」のどちらかに○をつけてください。

・3.5インチディスクを希望される方は，通信欄の「メディア」の「3.5」のほうに○をつけてください。

・通信欄には，「GENIE」やこの連載，CGAシステムや当チームについてのご意見，ご感想などをご自由にご記入ください。

・この振替用紙では，コンテストのビデオなどは申し込めません。マニュアルおよびバージョンアップだけです。

・この振替用紙で同時に複数申し込むことはできません。複数申し込まれる場合は郵便局に備えつけの振替用紙をご使用ください。

・CGAシステムは，当チームの活動の趣旨に賛同し，参加または協力の意思がある方にのみ使用が許され，営利目的には使用できません。あらかじめご了承ください。

・上記の費用は，マニュアルの印刷，発送などの実費であり，いわゆるプログラムの値段ではありません。プログラムの値段は基本的に無料ですが，当チームの活動に対するカンパは随時受けつけています。

・上記事項を守られていない方の入金は，ただの全額カンパとして処理されます。

ハードコア3Dエクスタシー(第10回)

SIDE A

ダブルバッファリングアニメーションの極意

Tan Akihiko 丹 明彦

**今回は「こいのぼりPRO-68K」のフォローとダブルバッファリングを解説していく
第2のステップへ進む前にちょっと寄り道することになるが
基本的なことなのでぜひ理解してもらいたい**

先月は思わぬ欠場となってしまったが，知らん顔をしつつ連載を再開する。

「バーチャレーシング」は，メガドライブ版で練習したおかげでようやくアーケード版の初級を完走することができた。なんといっても家庭用にはいくらでもやり直せる安心感があるので，コーナーも攻めるべきところは攻められるようになった。それにしてもアナログのハンドルとアクセルはいいものだと再確認した。

ようやくドリフトのしかたが少しずつわかってきた「デイトナUSA」であるが，これに採用されているシステムのうち，真似をしたいところがいくつかある。ひとつはタイムアタックモード。お邪魔カー一切なしで走りに専念できる。これは「デイトナ」のオリジナルではなく，「World Circuit」(AMIGA/IBM PC)や「Indianapolis 500: the simulation」(AMIGA/IBM PC)，それにメガドライブ版「バーチャレーシング」にも備えられているが，アーケードゲームで採用されたというのが素晴らしい。2つめはローリングスタート。前述の「Indy 500」ですでに採用されている方式だが，たとえばF-1の予選タイムアタックでは十分使える。3つめは壁にごりごりとぶつけながら走れる感覚。ストックカーゆえ当然ではあるのだが，ナムコの「リッジレーサー」で壁に当たっては失速，お邪魔カーに接触しては失速というのに，いらついていた私としては大歓迎だ。こうした気持ちいいドライビングゲームのための条件を備えたアーケードゲームが，少しずつでもいいから出てくることによって，一般ゲーマーがよいドライビングゲームについて見識を深めていくことを願ってやまない。

15fpsは夢ではない

さて，今年こそはゴールデンウィークにばりばりやろうと思っていたのだが，結局は高速道路風ドライブシミュレータに地味な改良を加えたにとどまった。が，ひとつ劇的なパフォーマンスの向上があり，X68030で15fpsを叩き出すことに成功した。ただし，この値は，「実数計算にコプロセッサを直接ドライブ」することによって達成したものである。マップシステムのようなものをCで書いたのだが，距離や方向を実数で計算していたのが予想外に重かったらしい。プログラムは変更せずに，gccのコンパイルオプションに「-m68040」をつけただけ。この結果をもって，「今後はコプロつきのX68030だけを対象にする」といいきることはさすがに許されない。マップシステムの計算精度は整数で十分であり，ましてテーブル化してしまえばほとんど無視できる計算量になるはずである。つまり実数計算というのは私のサボリなのである。ともあれ，15fpsは夢ではなくなった。ずいぶんと勇気づけられる事実である。

今月は補講の月

というわけで，今回も引き続き前進する予定でいた。お茶を濁しているわけにはいかない。もはや描画系は問題なく，車の力学的な運動に本気で取り組む時期にきているのだ。

実はドライブシミュレータの運転感覚が不自然なのを修正していたときに，“遠心力パラメータ”を導入しようとしている自分に気づいて愕然としたのである。いきあたりばったりのパラメータ導入とバランス調整。これでは，私自身がさんざん馬鹿にしていたラスターもののレーシングゲームがやっていることと少しも変わらないではないか。コーナリングはもっときちんとした理論によってシミュレートされるべきなのである。

以上の理由をもって今回は寄り道させていただきたい（1回休んでおいてこのいいぐさもなんだが)。技術的には問題はないが説明しておかなくてはならない重要な事項もある。内容が雑多になってしまう

ことをあらかじめお断りしたい。

「こいのぼり」へのフォロー

先月はゴールデンウィーク進行のために，付録ディスク「こいのぼりPRO-68K」に収録されたSLASH ver.2.0へのアフターケアを行うことができなかったので，遅ればせながらフォローする。

・GNU makeに通るmakeファイル

私はものぐさなので，SLASHライブラリの構築手順をmakeファイルに記述して，make一発ですべてコンパイルできるようにしている。階層構造になっているディレクトリをたどってmakeからさらにmakeを呼ぶようなこともやっている。この結果，現状ではシャープ純正のMAKE.Xでしかコンパイルできなくなってしまっている。

実はSLASHライブラリを作成する際に必要なツールのうち，シャープ純正のものはCOMMAND.XとこのMAKE.Xだけである。Cコンパイラもアセンブラもリンカもアーカイバも，Cライブラリさえもフリーソフトウェアやパブリックドメインソフトウェアで賄えるというのに。COMMAND.Xは標準でついてくるからいいとしても，MAKE.Xは「C Compiler PRO-68K」を買わないとついてこない。

やはりこれではまずいので，ソフトバンク刊行の「X68k Programming Series」で入手可能な環境でコンパイルできるようにmakeファイルを書き直した。X68000/030に移植されたGNU makeを通るmakeファイルを掲載しておく。

・slv2¥gMakefile　(リスト1)
・slv2¥core¥gMakefile　(リスト2)
・slv2¥stdcolor¥Makefile　(リスト3)
・slv2¥src¥gMakefile　(リスト4)

以上の4つを所定の位置に入れて，

```
gmake -f gMakefile
```

を実行するとよい。

なお，これらのmakeファイルはlibc本に収録されているGNU makeでのみ動作確認済み。また，同書に収録されているksh(パブリックドメインのKorn Shell)およびcp.x, rm.xが必要。健闘を祈る。

・旧FLOAT4.Xにご注意

X68030を数値演算コプロセッサ(68882)つきで使っている人への注意。サンプルの高速道路風ドライブシミュレータ(highway.x)をコンパイルする際には，実数演算ドライバにFLOAT4.X(68882を直接ドライブする)でなく，FLOAT2.X(ソフトウェアでエミュレーションする)を使ったほうが安全である。旧FLOAT4.Xを使うと，なぜか三角関数が腐るのである。私のX68030 Compactでも編集室のX68030でも確認した。しかし最新のHuman68k (SX-WINDOW ver.3.1システムキット)に付属するFLOAT4.Xでは改善されているため，症状が現れなくなった。入手することをお勧めしたい。

なお，旧バージョンでも，コンパイルをFLOAT2.Xを組み込んだシステムで行えば，実行はFLOAT4.Xを組み込んでもかまわないようだ。

・その他

コンパイルできないというお便りを数件いただいたが，いずれも当方では再現できなかった。申しわけない。

ダブルバッファリング詳説

少しは技術の香りがする話題もしておこう。ダブルバッファリング(double buffering)とは，ちらつきのないアニメーションを実現するための手法である。画面とは別に画面と同じバッファを確保しておき，描画はそのバッファに対して行う。描画が終了したら，バッファと画面を入れ替える。これを繰り返すことで，描画の過程を画面に出すことなくアニメーションを行うことができるのである。

言葉でいうと難しい。図1をご覧いただきたい。(A)がダブルバッファリングを行わなかった場合の画面の様子である。ここでは描画作業はすべて表示されている画面で行われる。つまり，前回描画した画像を消去し，新しい画像を描画する，その過程がすべて画面に表示されている。このため，画面は点滅しているかのようにちらつき，非常に見苦しい。

図1(B)はダブルバッファリングを導入したあとの様子である。描画作業は裏ページに対して行うため，画面の表示はちらつくことがない。

ここで裏ページとはバッファのこと。X68000/030ではVRAMの空きページをバッファに用いることでアニメーションを円滑に行える。ページ切り替え(実際はハードウェアスクロール)によって，バッファと表示画面の交換を一瞬で行うことができるのだ。

さらに，X68000/030では割り込みを用いて垂直帰線期間にページ切り替えを行うことができる。これには，美しくページ切り替えができ，またページ切り替えを一定周期で行うことができるという2つの効果がある。こうした機能を生かせるのは，ハードウェアに近い部分を比較的容易に制御できるX68000/030の強みといえるだろう。

さて，今回掲載するダブルバッファリングのためのライブラリの動作について解説しよう。アプリケーションプログラマはこのライブラリの動作の詳細を知っておく必要はないと思うが，知っておいたほうがきちんと動くアプリケーションを書きやすい。

図2(A)がダブルバッファリングに用いられるページ構成のイメージである。表示画面256×256ドッ

ト，実画面512×512ドットモードのグラフィック画面上に2ページのバッファを確保して，交互に表示し，表示しないほうのページに描画を行う。

図2(B)と(C)が動作タイミングチャートのようなものである。少しわかりづらいが，横方向が時間の流れ，縦の線は垂直帰線割り込み周期(モニタのモードが15kHzの場合で約1/60秒)，縦の太い線はページ切り替え周期(startDoubleBuffer()の引数で指定する，図2(B)と(C)は4を指定した場合)，2本の横方向の帯はダブルバッファリングの2つのバッファの状態を示す。

図2(B)は描画が十分に高速という状況のもとで起こる，理想的な処理の流れである。1周期の間にアプリケーションのやるべきことは4つある。

・クリア …… 現在のページの内容(2周期前に描画されて1周期前に表示された)を消去する(ClearBox())

・ドロー …… 座標変換(TranslateAll())および描画(DisplayPolygonList())を行う

・トリガ …… 描画が終了したのでページ切り替えしてよいということをシステムに知らせる(toggleDoubleBufferApage())

・ウェイト …… ページ切り替えが行われるまで待つ(waitDoubleBufferSync)

注意していただきたいのは，「ページ切り替えをアプリケーションで行ってはいけない」ということである。ページ切り替えは割り込みルーチンが，しかるべきタイミングで行うのである。その条件は，

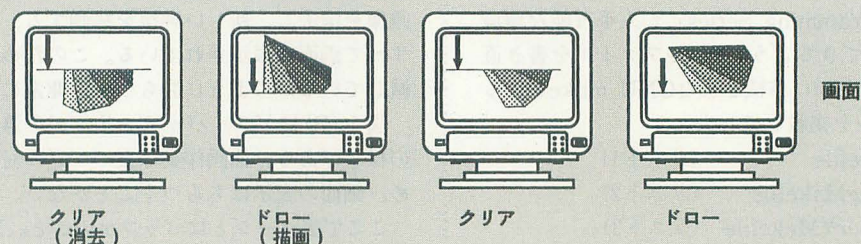
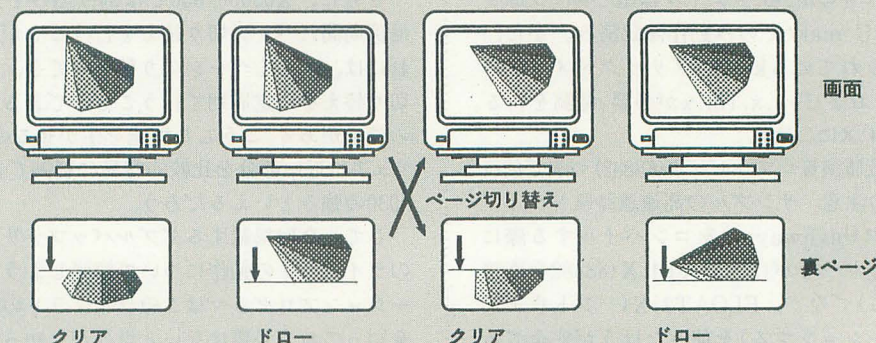
1)　垂直帰線期間中であること

2)　内部カウンタ(ページ切り替え周期を制御するカウンタ)が0であること

3)　アプリケーション側からページ切り替えの許可が出ていること

の3つである。1)は垂直帰線割り込みで入ってくるルーチンのことであるから常に成立する。2)と3)は説明が必要だろう。これはアプリケーションと割り込みルーチンの間で同期を取っているのである。「内部カウンタ」とはページ切り替え直後に値がページ切り替え周期の長さにリセットされ，割り込みごとに1減算される。つまり，描画が速すぎてもページ切り替え周期がくるまで待たせることができる。

こうした「同期取り」の必要性は，「描画処理は

図1　ダブルバッファリングの必要性

常に一定時間で終了するとは限らない」という事実にもとづいている。もし描画処理の所要時間が常にある範囲内に収まることが保証されるなら，割り込みルーチンはなにも考えずにページを切り替えればよい。しかしポリゴンの描画時間は一定ではない。速すぎるぶんにはまだいいのだが，困ったことにポリゴンの描画はしばしば遅れが生じるものなのである。それが図2(C)の状況である。ページ切り替え周期がやってきたのにポリゴン描画が終わっていない。ここでページ切り替えをやってしまっては，不完全な画像を表示してしまう。

というわけで描画が遅れた場合，割り込みルーチン側では，条件3)のアプリケーションからのページ切り替え許可がないのでページ切り替えを行わない。そして“サドンデス”期間に入る。ここから先は内部カウンタを0のままホールドし，アプリケーションからページ切り替え許可が出たら次の割り込みで即座にページを切り替える。

妙な動作と思うかもしれないが，この仕様になるまでにはかなりの試行錯誤があったのだ。おかげで，どんな状況のもとでも最も滑らかな操作感覚を得ることができたと自負している。

このライブラリを使用するためには，プログラムを入力して所定の場所に置き，再コンパイルすればいい。「こいのぼりPRO-68K」に収録した同名のファイルには，バグというか仕様に欠陥があった。

・slv2¥src¥doublebuffer.s　（リスト5）
・slv2¥include¥slash2¥doublebuffer.h　（リスト6）

このダブルバッファリング用ライブラリはSLASHとは独立である。表示画面256×256ドット，実画面512×512ドットモードでしか使えない固定仕様であるが，改造そのほかは自由としておく。親切なプログラムではないし，使い方を誤ると簡単に誤動作するので，リスクをふまえてお使いいただきたい。

サンプルプログラム

ダブルバッファリングを効果的に使うSLASHプログラミングのサンプルとして，画面中央をブリリアントカットされた石が回転するだけのプログラムを書いてみた。SLASHを利用したC言語によるプログラムとしては，最低限の要素しか入っていない。そして，私がこれまで書いたSLASHアプリケーションはすべて，このプログラムを基本構造として肉づけしていったものである。

解析を試みるのであれば，以下のものに注意を払ってみていただきたい。

・**SLASH標準ファンクションコール**

AddNorm()
SetClearPlane()
ClearBox()
SetWritePlane()
TranslateAll()
DisplayPolygonList()
AdjustMinMax()

・**形状作成用簡易関数**

addtriangle()
addtetragon()

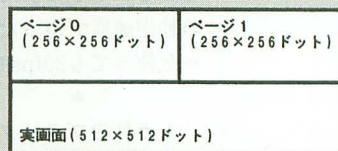
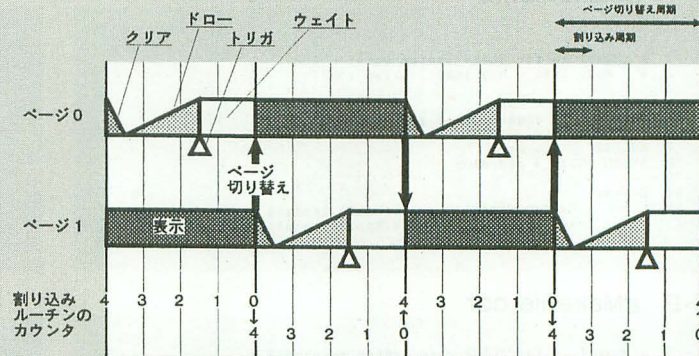
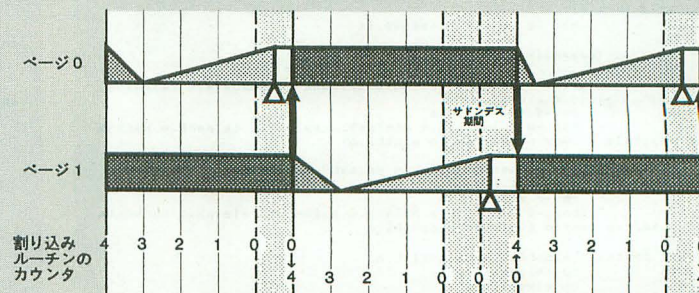
・**ダブルバッファリング**

setDoubleBufferMode()
startDoubleBuffer()
waitDoubleBufferSync
toggleDoubleBufferApage()

・**パフォーマンスモニタ(fps値の測定に用いる)**

PMreset
PMcount
PMaverage

・**時間差測定**

図2　ダブルバッファリングの動作

TIMEDIFFERENCE()

プログラムの作成方法は，以下のプログラムを入力して同じディレクトリに置き，makeを実行する。

・Makefile　　(リスト7)
・diamond.h　　(リスト8)
・diamond.c　　(リスト9)
・jewel.c　　(リスト10)

今回のプログラムは，モデルが簡単なせいもあるが，結構高速に動作する。ちなみに実行速度は，

・X68000　　15fps
・X68000XVI　　20fps
・X68030　　30fps

である。それぞれダブルバッファリングの開始時にページ切り替え周期(startDoubleBuffer()関数の引数)を4,3,2に設定して得られた値だ。これより周期を短くすると処理落ちが起こる(見ているぶんにはわからない)。時間を管理しているので，速いマシンほど滑らかに動く。回転速度は変わらない。

なお，ダブルバッファリング(というより割り込み)を使わないで実行すると，正味の動作速度を計ることができる。その場合はX68030だと45fps相当の動作速度であった。つまり，もう少し複雑なモデルを使っても30fpsで動かすことは可能だろう。

＊　　＊　　＊

1カ月のブランクでできた時間を車の力学法則の煮詰めに使おうと考えていたのだが，魔がさしてテクスチャマップもどきに手をつけ，のめりこんでしまった。えてして道草というのは面白い。

レース雑誌からF-1マシンの写真をスキャナで取り込んでポリゴンに張りつけてみると，ポリゴン数の割にはリアルになる。全部C言語で書いたにもかかわらず，5～8fps程度は表示できる。高画質モードまで作ったら，さすがに遅くなった。

まあ，私のやっているのはテクスチャマップとはとても呼べないシロモノではある。自由変形であるから，遠近の差が激しい部分で破綻するのだ。が，動いていればあまりわからない。あのグラフィックワークステーション業界の雄，シリコングラフィックス社のマシンでも数年前まではその程度の仕様だったらしい(マニュアルにはご丁寧にも破綻の回避方法まで書いてある)。

このへんになるとアルゴリズムもだいたい固定してくるので，いっそのことハードウェア化したくなってくる。I/Oスロットに入るグラフィックボード。出力画像をイメージ端子から入力してX68000/030のG-RAMに表示すれば，どんなに複雑な画像でも30fpsは出せるのだ。もはやシャープも見捨てた感のあるイメージ端子だが，こういう使い道は見いだせないものだろうか。

積み残し事項を集めてフォローしていたらいつのまにか脇道にそれそうになってしまったが，また車ゲームに戻っていけるだろう。

リスト1　gMakefile.lib

```
 1: # gMakefile for SLASH system version 3.0
 2: #   Aug. 1993 - May 1994   A.Tan (Oh!X)
 3:
 4: SHELL       = ksh.x
 5: MAKE        = gmake.x -f gMakefile
 6: COREDIR     = core
 7: SRCDIR      = src
 8: STDCOLORDIR = stdcolor
 9:
10: depend:
11:         cd $(COREDIR)     ; $(MAKE) install ; cd ..
12:         cd $(SRCDIR)      ; $(MAKE) install ; cd ..
13:         cd $(STDCOLORDIR) ; $(MAKE) install ; cd ..
14:
15: clean:
16:         cd $(COREDIR)     ; $(MAKE) clean ; cd ..
17:         cd $(SRCDIR)      ; $(MAKE) clean ; cd ..
18:         cd $(STDCOLORDIR) ; $(MAKE) clean ; cd ..
19:
20: distclean:
21:         rm -f slashlib.a tslashlib.a _slashlib.a stdcolor.a
22:         cd $(COREDIR)     ; $(MAKE) distclean ; cd ..
23:         cd $(SRCDIR)      ; $(MAKE) distclean ; cd ..
24:         cd $(STDCOLORDIR) ; $(MAKE) distclean ; cd ..
```

リスト2　gMakefile.cor

```
 1: # gMakefile for SLASH core system version 3.0
 2: #   May 1994   T.Yokouchi (Oh!X)
 3:
 4: SHELL = ksh.x
 5:
 6: %.o: %.s
 7:         has -z -u -w -n -m5000 $<
 8:
 9: all: tslashlib.a slashlib.a
10:
11: tslashlib.a: atslash.o tslash.o tslashR.o bankdata.o rayshade.
o xway.o zwayroad.o sintbl.o
12:         rm -f tslashlib.a
13:         har -u tslashlib.a atslash.o tslash.o tslashR.o bankda
ta.o rayshade.o xway.o zwayroad.o sintbl.o
14:
15: slashlib.a: aslash.o gslash.o gslashR.o bankdata.o rayshade.o
xway.o zwayroad.o sintbl.o
16:         rm -f slashlib.a
17:         har -u slashlib.a aslash.o gslash.o gslashR.o bankdata
.o rayshade.o xway.o zwayroad.o sintbl.o
18:
19: install: slashlib.a tslashlib.a
20:         cp tslashlib.a ..
21:         cp slashlib.a ..
22:
23: atslash.o: aslash.s
24:         has -s __TEXT__ -s __xMULT8__ -z -u -w -n -m5000 -o at
slash.o aslash.s
25:
26: aslash.o: aslash.s
27:         has -s __xMULT8__ -z -u -w -n -m5000 aslash.s
28:
29: gslashR.o: gslash.s
30:         has -s __RASTER__ -z -u -w -n -m5000 -o gslashR.o gsla
sh.s
31:
32: tslashR.o: tslash.s
33:         has -s __RASTER__ -z -u -w -n -m5000 -o tslashR.o tsla
sh.s
34:
35: gslash.o: gslash.s
36: tslash.o: tslash.s
37: bankdata.o: bankdata.s
38: rayshade.o: rayshade.s
39: xway.o: xway.s
40: zwayroad.o: zwayroad.s
41: sintbl.o: sintbl.s
42:
43: clean:
44:         rm -f atslash.o tslash.o tslashR.o
45:         rm -f aslash.o gslash.o gslashR.o
46:         rm -f bankdata.o rayshade.o xway.o zwayroad.o sintbl.o
47:
48: distclean: clean
49:         rm -f tslashlib.a slashlib.a
```


▶7月号の付録ディスクはX68030＋MC68882の速さが実感できてとてもよかったです(笑)。なにせいままではつけてあっただけですから。　　小川　克仁(20)愛知県

リスト3 gMakefile.std

```
 1: # gMakefile for _slash standard color library version 3.0
 2: #     Aug. 1993 - May 1994    A.Tan (Oh!X)
 3:
 4: SHELL = ksh.x
 5:
 6: %.o: %.s
 7:         has -u -w $<
 8:
 9: all: stdcolor.a
10:
11: install: stdcolor.a
12:         cp stdcolor.a ..
13:
14: stdcolor.a: standard.o black.o gray.o gold.o silver.o rainbow.
o¥
15:           darkblue.o blue.o darkred.o red.o orange.o yellow.o
16:         rm -f stdcolor.a
17:         har -u stdcolor.a standard.o black.o gray.o gold.o sil
ver.o rainbow.o¥
18:                         darkblue.o blue.o darkred.o red.o or
ange.o yellow.o
19:
20: standard.o: standard.s
21: black.o: black.s
22: gray.o: gray.s
23: gold.o: gold.s
24: silver.o: silver.s
25: rainbow.o: rainbow.s
26: darkblue.o: darkblue.s
27: blue.o: blue.s
28: darkred.o: darkred.s
29: red.o: red.s
30: orange.o: orange.s
31: yellow.o: yellow.s
32:
33: clean:
34:         rm -f standard.o black.o gray.o gold.o silver.o rainbo
w.o¥
35:                 darkblue.o blue.o darkred.o red.o orange.o yello
w.o
36:
37: distclean: clean
38:         rm -f stdcolor.a
```

リスト4 gMakefile.src

```
 1: # gMakefile for _slash utility library version 3.0
 2: #     Aug. 1993 - May 1994    A.Tan (Oh!X)
 3:
 4: SHELL = ksh.x
 5:
 6: CCOPTS = -O -Wall -I../include
 7: #CCOPTS = -fno-defer-pop -O -Wall
 8:
 9:
10: %.o: %.s
11:         has -u -w $<
12:
13: %.o: %.c
14:         gcc $(CCOPTS) -c $<
15:
16:
17: all: _slashlib.a
18:
19:
20: install: _slashlib.a
21:         cp _slashlib.a ..
22:
23:
24: _slashlib.a: _aslash.o _atslash.o _gslash.o _grslash.o _tslash
.o _trslash.o sortpoly.o addprim.o euler.o euler882.o xfer.o xferR.o c
lear.o clearR.o doublebuffer.o perfmon.o slmath.o check.o viewer.o
25:         rm -f _slashlib.a
26:         har -u _slashlib.a¥
27:                 _aslash.o _atslash.o _gslash.o _grslash.o _tsl
ash.o _trslash.o¥
28:                 sortpoly.o addprim.o euler.o euler882.o xfer.o
 xferR.o clear.o clearR.o doublebuffer.o perfmon.o slmath.o check.o vi
ewer.o
29:
30: _aslash.o: _aslash.s
31:
32: _atslash.o: _aslash.s
33:         has -u -w -s __TEXT__ -o _atslash.o _aslash.s
34:
35: _gslash.o: _gslash.s
36:
37: _grslash.o: _gslash.s
38:         has -u -w -s __RASTER__ -o _grslash.o _gslash.s
39:
40: _tslash.o: _gslash.s
41:         has -u -w -s __TEXT__ -o _tslash.o _gslash.s
42:
43: _trslash.o: _gslash.s
44:         has -u -w -s __TEXT__ -s __RASTER__ -o _trslash.o _gsl
ash.s
45:
46: sortpoly.o: sortpoly.c
47:         gcc $(CCOPTS) -c $<
48: #       gcc $(CCOPTS) -DSTRATEGY1 -DDEBUG -c $<
49:
50: addprim.o: addprim.c
51: viewer.o: viewer.c
52: check.o: check.c
53: euler.o: euler.c
54: perfmon.o: perfmon.c
55: slmath.o: slmath.c
56: xfer.o: xfer.s
57: clear.o: clear.s
58: doublebuffer.o: doublebuffer.s
59:
60: euler882.o: euler.c
61:         gcc $(CCOPTS) -m68040 -DFPU882 -o euler882.o -c $<
62:
63: xferR.o: xfer.s
64:         has -u -w -s __RASTER__ xfer.s -o xferR.o
65:
66: clearR.o: clear.s
67:         has -u -w -s __RASTER__ clear.s -o clearR.o
68:
69: clean:
70:         rm -f _aslash.o _atslash.o _gslash.o _grslash.o _tslas
h.o _trslash.o¥
71:                 sortpoly.o addprim.o euler.o euler882.o xfer.o
 xferR.o clear.o¥
72:                 clearR.o doublebuffer.o perfmon.o slmath.o che
ck.o viewer.o
73:
74: distclean: clean
75:         rm -f _slashlib.a
```

リスト5 doublebuffer.s

```
 1: *       doublebuffer.s
 2: *       - SLASHアプリケーションのサポートルーチン
 3: *         垂直帰線期間割り込みによって
 4: *         ちらつきのないページ切り替えを実現する
 5: *         ダブルバッファリング・アニメーションに必要な処理
 6: *         Oct. 1993 - Apr. 1994
 7: *               丹 明彦(Oh!X)
 8:
 9: *       version 1.0     Oct. 1993
10: *       version 2.1     Apr. 1994
11: *       version 2.2     May  1994
12:
13: *(history)
14: *       1993/10/11      バージョン1.0
15: *       1994/4/20       テキストVRAMのページ切り替えを追加
16: *                       テキスト/グラフィックのモードをセットするコールを追加
17: *       1994/5/5        doubleBuffer[AV]pageを外部参照
18: *                       (アルゴリズムの誤りを正すため)
19: *                       toggleDoubleBufferVpage()のバグを修正
20: *                       (doubleBufferMode[GT]の意味が逆になっていた)
21: *                       startDoubleBuffer()に割り込み周期を制御するため
のカウント指定を追加
22: *       1994/5/18       割り込み周期の処理を変更した
23: *                       (変更前) 割り込みはcount/60秒ごと,
24: *                                ページ切り替えはn*count/60秒ごと
25: *                       (変更後) 割り込みは1/60秒ごと,
26: *                                ページ切り替えは(n+cycle)/60秒ごと
27: *       1994/5/18       ダブルバッファリング開始時にページ1に切り替える
28:
29:
30:         .xdef   _setDoubleBufferMode
31:         .xdef   _startDoubleBuffer
32:         .xdef   _endDoubleBuffer
33:         .xdef   _toggleDoubleBufferApage
34: *       .xdef   _toggleDoubleBufferVpage
35:
36:         .xdef   _doubleBufferModeG
37:         .xdef   _doubleBufferModeT
38:         .xdef   _doubleBufferApage
39:         .xdef   _doubleBufferVpage
40:
41:
42:         .include        IOCSCALL.MAC
43:
44:
45: _doubleBufferModeG:     dc.l    1
46: _doubleBufferModeT:     dc.l    0
47:
48: _doubleBufferCycle      dc.l    1
49: _doubleBufferCount      dc.l    1
50:
51: _doubleBufferVpage:     dc.l    256
52: _doubleBufferApage:     dc.l    256
53:
54:
55: *void   setDoubleBufferMode( int gmode, int tmode )
56: *       - グラフィックVRAMとテキストVRAMの一方または両方で
57: *         ダブルバッファリングを行うよう設定する
58: *         gmode, tmodeともに≠0でダブルバッファリングを行う
59: *         デフォルトはグラフィックVRAMのみダブルバッファリングを行う
60:
61: _setDoubleBufferMode:
62:         move.l  4(sp),_doubleBufferModeG        * gmode
63:         move.l  8(sp),_doubleBufferModeT        * tmode
64:         rts
65:
```

▶ディスプレイが壊れ，修理に出しました。修理中にはX68000のない生活を送ったことになりますが，あらためて自分の生活にX68000が溶け込んでいることに気づきました。
一ノ瀬　宣彦(23)東京都

```
    66:
    67: *           int startDoubleBuffer( int cycle )
    68: *           - ダブルバッファリングのための割り込みを設定する。
    69: *             垂直帰線期間ごとに割り込みを発生させ
    70: *             cycle回以上ごとにページ切り替えを行う。
    71: *             これにより,
    72: *             (1) 描画の速かったフレームではcycle/60秒経過するまで待ち,
    73: *             (2) 描画の遅かったフレームでは速やかに切り替える
    74: *             という処理が可能になる。
    75: *             戻り値はIOCSコールのVDISPSTのそれと同じ
    76: *             (0:成功, ≠0:すでに割り込み使用中)
    77:
    78: _startDoubleBuffer:
    79:             move.l  d1,-(sp)
    80:
    81:             move.l  1*4+4(sp),d0                * cycle
    82:             move.l  d0,_doubleBufferCycle
    83:             move.l  d0,_doubleBufferCount
    84:
    85:             move.l  #256,d2                     * 表示位置のx座標
    86:             move.l  d2,_doubleBufferVpage
    87:             move.l  d2,_doubleBufferApage
    88:
    89:             moveq.l #0,d3                       * 表示位置のy座標
    90:
    91:                                                 * 表示ページを初期状態(ページ1
)に
    92:             move.l  _doubleBufferModeG,d1       * グラフィックVRAM
    93:             beq     sdb2
    94:             move.l  d3,d1                       * ページ(グラフィックVRAMの有
効ページすべて)
    95:             IOCS    __HOME                      * グラフィックVRAMのページ切り
替え
    96: sdb2:
    97:             move.l  _doubleBufferModeT,d1       * テキストVRAM
    98:             beq     sdb1
    99:             moveq.l #8,d1                       * モード(テキストの検査)
   100:             IOCS    __SCROLL                    * テキストVRAMのページ切り替え
(d2,d3は共通)
   101: sdb1:
   102:
   103:             lea.l   _toggleDoubleBufferVpage,a1
   104:             move.w  #00_01,d1                         * 垂直帰線期間で毎回
割り込む
   105:             IOCS    __VDISPST
   106:
   107:
   108:
   109:
   110:
   111:
   112:
   113:
   114:             move.l  (sp)+,d1
   115:             rts
   116:
   117:
   118: *           int endDoubleBuffer()
   119: *           - ダブルバッファリングのための割り込みを解除する
   120: *             戻り値はIOCSコールのVDISPSTのそれと同じ
   121:
   122: _endDoubleBuffer:
   123:             move.l  d1,-(sp)
   124:             lea.l   0,a1                * アドレスに0を与えれば割り込み禁止
   125:             move.w  #00_01,d1           * 必要ないとは思うが念のため
   126:             IOCS    __VDISPST
   127:             move.l  (sp)+,d1
   128:             rts
   129:
   130:
   131: *           int toggleDoubleBufferApage()
   132: *           - アクティブページを切り替える
   133: *             すなわち, 次の VDISP 割り込みで
   134: *             表示ページを切り替えることを許可する
   135: *             メインルーチン側から, 新しいページの
   136: *             描画を終えた時点で呼び出す
   137: *             戻り値 =0 : 成功
   138: *                    ≠0 : 前回の表示ページ切り替えが完了していない
   139:
   140: _toggleDoubleBufferApage:
   141:             move.l  d1,-(sp)
   142:             move.l  _doubleBufferVpage,d0
   143:             cmp.l   _doubleBufferApage,d0
   144:             bne     tdba1
   145:             move.l  #256,d1                     * apage = 256 - apage
   146:             sub.l   d0,d1
   147:             move.l  d1,_doubleBufferApage
   148:             moveq.l #0,d0
   149:             bra     tdba2
   150: tdba1:      moveq.l #1,d0
   151: tdba2:      move.l  (sp)+,d1
   152:             rts
   153:
   154:
   155: *           void toggleDoubleBufferVpage()
   156: *           - 表示ページを切り替える
   157: *             VDISP 割り込みルーチンなので
   158: *             直接呼び出されることはない
   159:
   160: _toggleDoubleBufferVpage:
   161:             movem.l d0-d3,-(sp)
   162:
   163:             move.l  _doubleBufferCount,d0       * doubleBufferCycle回以
上経過することが必要
   164:             beq     tdbv4                       * いわゆるサドンデス状態
   165:             subq.l  #1,d0                       * カウントダウン
   166:             move.l  d0,_doubleBufferCount
   167:             bne     tdbv1                       * まだdoubleBufferCycle
回に達していない
   168:
   169: tdbv4:
   170:             move.l  _doubleBufferApage,d2       * アクティブページ = 表示位置
のx座標
   171:             cmp.l   _doubleBufferVpage,d2       * 現在の表示位置
   172:             beq     tdbv1                       * 両者が等しい == ページ切り替
え許可が出ていない
   173:
   174:                                                 * ページ切り替えを行う条件が揃っ
た
   175:             move.l  _doubleBufferCycle,_doubleBufferCount    * カウン
タのリセット
   176:
   177:             moveq.l #0,d3                       * 表示位置のy座標
   178:
   179:             move.l  _doubleBufferModeG,d1
   180:             beq     tdbv3
   181:             move.l  d3,d1                       * ページ(グラフィックVRAMの有
効ページすべて)
   182:             IOCS    __HOME                      * グラフィックVRAMのページ切り
替え
   183: tdbv3:
   184:             move.l  _doubleBufferModeT,d1
   185:             beq     tdbv2
   186:             moveq.l #8,d1                       * モード(テキストの検査)
   187:             IOCS    __SCROLL                    * テキストVRAMのページ切り替え
(d2,d3は共通)
   188: tdbv2:
   189:             move.l  d2,_doubleBufferVpage       * 現在の表示位置を更新
   190: tdbv1:      movem.l (sp)+,d0-d3
   191:             rte
```

リスト6 doublebuffer.h

```
     1: /*
     2:             doublebuffer.h
     3:             - SLASHアプリケーションのサポートルーチン
     4:               垂直帰線期間割り込みによって
     5:               ちらつきのないページ切り替えを実現する
     6:               ダブルバッファリング・アニメーションに必要な処理
     7:               Oct. 1993 - May 1994
     8:                     丹 明彦(Oh!X)
     9:
    10:             version 1.0     Oct. 1993
    11:             version 2.1     Apr. 1994
    12:             version 2.2     May  1994
    13:
    14: (history)
    15:             1993/10/11      バージョン1.0
    16:             1994/4/20       テキストVRAMのページ切り替えを追加
    17:                             テキスト/グラフィックのモードをセットするコールを追加
    18:             1994/5/5        テキスト/グラフィックのモードの意味が逆転していたバグを
修正
    19:                             ページをC言語のレベルで参照できるようにした
    20:                             ページ切り替えが行われたことを確認するマクロを追加
    21:                             startDoubleBuffer()に割り込み周期を制御するため
のカウント指定を追加
    22:             1994/5/18       割り込み周期の処理を変更した
    23:                             (変更前) 割り込みはcount/60秒ごと,
    24:                                      ページ切り替えはn*count/60秒ごと
    25:                             (変更後) 割り込みは1/60秒ごと,
    26:                                      ページ切り替えは(n+cycle)/60秒ごと
    27:             1994/5/18       ダブルバッファリング開始時にページ1に切り替える
    28: */
    29:
    30: #ifndef __DOUBLEBUFFER_H__
    31: #define __DOUBLEBUFFER_H__
    32:
    33:
    34: /* グラフィックまたはテキストVRAMでダブルバッファをするかしないか */
    35: extern unsigned long    doubleBufferModeG;
    36: extern unsigned long    doubleBufferModeT;
    37:
    38: /* ページ切り替え最低周期とそのためのカウンタ(アプリケーション側からのアクセス不要) */
    39: /*
    40: extern unsigned long    doubleBufferCycle;
    41: extern unsigned long    doubleBufferCount;
    42: */
    43:
    44: /* アプリケーション側から表示を許可したページ */
    45: extern unsigned long    doubleBufferApage;
    46:
    47: /* 実際に表示されているページ(割り込みルーチンが変更する) */
    48: extern unsigned long    doubleBufferVpage;
    49:
    50:
    51: void    setDoubleBufferMode( int gmode, int tmode );
    52: /*      - グラフィックVRAMとテキストVRAMの一方または両方で
    53:           ダブルバッファリングを行うよう設定する
    54:           gmode, tmodeともに≠0でダブルバッファリングを行う
    55:           デフォルトはグラフィックVRAMのみダブルバッファリングを行う      */
    56:
    57: int     startDoubleBuffer( int cycle );
    58: /*      - ダブルバッファリングのための割り込みを設定する
    59:           垂直帰線期間ごとに割り込みを発生させ
    60:           cycle回以上ごとにページ切り替えを行う。
    61:           これにより,
    62:           (1) 描画の速かったフレームではcycle/60秒経過するまで待ち,
    63:           (2) 描画の遅かったフレームでは速やかに切り替える
    64:           という処理が可能になる。
    65:           戻り値はIOCSコールのVDISPSTのそれと同じ
    66:           (0:成功, ≠0:すでに割り込み使用中)      */
    67:
    68: int     endDoubleBuffer();
```


▶理由は聞かないでください。2週間ほど仕事を休んで北海道へ旅立ちます。
原田 健史(26)東京都

```
69: /*        - ダブルバッファリングのための割り込みを解除する
70:             戻り値はIOCSコールのVDISPSTのそれと同じ          */
71:
72: int       toggleDoubleBufferApage();
73: /*        - アクティブページを切り替える
74:             すなわち，次の VDISP 割り込みで
75:             表示ページを切り替えることを許可する
76:             メインルーチン側から，新しいページの
77:             描画を終えた時点で呼び出す
78:             戻り値 =0 : 成功
79:                    ≠0 : 前回の表示ページ切り替えが完了していない */
80:
81: /* void toggleDoubleBufferVpage()
```

```
   82:            - 表示ページを切り替える
   83:              VDISP 割り込みルーチンなので
   84:              直接呼び出されることはない      */
   85:
   86: #define checkDoubleBufferSync
(doubleBufferApage == doubleBufferVpage)
   87: /*         - ページ切り替えが行われたかどうかチェックする */
   88:
   89: #define waitDoubleBufferSync
 while(doubleBufferApage != doubleBufferVpage)
   90: /*         - ページ切り替えが行われるまで待つ */
   91:
   92: #endif  /* __DOUBLEBUFFER_H__ */
```

リスト7 Makefile

```
 1: # Makefile for jewel.x
 2: # ハードコア3Dエクスタシー SIDE A
 3: # Oh!X 1994年7月号
 4:
 5: SHELL              = a:¥command.x
 6: SLASHLIBDIR        = ..¥slv2
 7: SLASHINCLUDEDIR    = ..¥slv2¥include
 8: CCOPTS             = -O -Wall -I$(SLASHINCLUDEDIR)
 9:
10: %.o: %.c
11:         gcc $(CCOPTS) -c $<
12:
13: all: jewel.x
14:
15: jewel.x: jewel.o diamond.o
16:         gcc -o jewel.x jewel.o diamond.o¥
17:                   $(SLASHLIBDIR)¥stdcolor.a¥
18:                   $(SLASHLIBDIR)¥_slashlib.a¥
19:                   $(SLASHLIBDIR)¥slashlib.a
20:
21: jewel.o: jewel.c diamond.h
22: diamond.o: diamond.c diamond.h
23:
24: clean:
25:         if exist jewel.o del jewel.o
26:         if exist diamond.o del diamond.o
27:
28: distclean: clean
29:         if exist jewel.x del jewel.x
```

リスト8 diamond.h

```
 1: /*
 2:          diamond.h
 3:          - ブリリアントカットのようなオブジェクトを作る
 4:            May 1994        丹 明彦(Oh!X)
 5: */
 6:
 7: #ifndef __DIAMOND_H__
 8: #define __DIAMOND_H__
 9:
10: #include        <slash2/slashlib.h>
11:
12: /* 頂点数: 8*4+1 */
13: #define DIAMOND_NPOINT              33
14: /* ポリゴン数: 8*7+3
15:      57面カットだがSLASHの都合で上面を3分割した */
16: #define DIAMOND_NPOLYGON           59
17:
18:
19: #ifndef DIAMOND_BODY
20:
21: /* ヘッダファイルだけに有効な外部参照宣言
22:      (diamond.cでは本体を宣言している) */
23:
24: extern SLPOLYGONLIST     *diamond_polygonlist;
25: extern SLPOINTLIST       *diamond_pointlist;
26:
27: void     createDiamond();
28: void     destroyDiamond();
29:
30: #endif  /* DIAMOND_BODY */
31:
32: #endif  /* __DIAMOND_H__ */
```

リスト9 diamond.c

```
 1: /*
 2:          diamond.c
 3:          - ブリリアントカットのようなオブジェクトを作る
 4:            May 1994        丹 明彦(Oh!X)
 5: */
 6:
 7: #include        <stdlib.h>
 8: #include        <math.h>
 9:
10: #include        <slash2/slashlib.h>
11: #include        <slash2/addprim.h>
12: #include        <slash2/stdcolor.h>
13:
14: /* ポリゴンリストと頂点リスト */
15: SLPOLYGONLIST    *diamond_polygonlist;
16: SLPOINTLIST      *diamond_pointlist;
17:
18: /* 諸定数を定義する */
19: #define DIAMOND_BODY
20: #include        "diamond.h"
21:
22: /* 各頂点の高さ */
23: #define H0      -45
24: #define H1      -40
25: #define H2      -20
26: #define H3      0
27: #define H4      100
28:
29: /* 各高さに対応する半径 */
30: #define R0      70
31: #define R1      90
32: #define R2      110
33: #define R3      100
34: #define R4      0
35:
36: /* 頂点座標 */
37: static int      x[5][8], y[5][8], z[5][8];
38:
39: /* 頂点番号から座標を得るマクロ
40:      円周方向にはループするので8で割った余りを使う */
41: #define V(H,T)  x[H][(T)%8],y[H][(T)%8],z[H][(T)%8]
42:
43: /* 色は標準色ライブラリのものを用いる */
44: #define DIAMOND_COLOR    gray
45:
46: /* 形状を作成する */
47: void    createDiamond()
48: {
49:         int      i;
50:         double   t1, t2;
51:
```

```
   52:           /* 領域確保と初期化 */
   53:           diamond_polygonlist = malloc( sizeof(SLPOLYGONLIST)+si
zeof(SLPOLYGON)*DIAMOND_NPOLYGON );
   54:           diamond_polygonlist->n = 0;
   55:           diamond_pointlist = malloc( sizeof(SLPOINTLIST)+sizeof
(SLPOINT)*DIAMOND_NPOINT );
   56:           diamond_pointlist->n = 0;
   57:
   58:           /* 頂点座標をあらかじめ計算しておく */
   59:           for ( i = 0; i < 8; i++ ) {
   60:                   t1 = (      i*45.0)*M_PI/180.0;  /* 8角形 */
   61:                   t2 = (22.5+i*45.0)*M_PI/180.0;  /* 半分ずらした8角
形 */
   62:                   x[0][i] = R0*cos(t1); y[0][i] = H0; z[0][i] =
R0*sin(t1);
   63:                   x[1][i] = R1*cos(t2); y[1][i] = H1; z[1][i] =
R1*sin(t2);
   64:                   x[2][i] = R2*cos(t1); y[2][i] = H2; z[2][i] =
R2*sin(t1);
   65:                   x[3][i] = R3*cos(t2); y[3][i] = H3; z[3][i] =
R3*sin(t2);
   66:                   x[4][i] = R4;          y[4][i] = H4; z[4][i] =
R4;
   67:           }
   68:           /* 側面……8×7=56面 */
   69:           for ( i = 0; i < 8; i++ ) {
   70:                   /* 計算しておいた頂点座標を拾って3角形を登録していく */
   71:                   addtriangle( diamond_polygonlist, diamond_poin
tlist, V(0,i), V(0,i+1), V(1,i  ), &DIAMOND_COLOR );
   72:                   addtriangle( diamond_polygonlist, diamond_poin
tlist, V(1,i), V(0,i+1), V(1,i+1), &DIAMOND_COLOR );
   73:                   addtriangle( diamond_polygonlist, diamond_poin
tlist, V(1,i), V(1,i+1), V(2,i+1), &DIAMOND_COLOR );
   74:                   addtriangle( diamond_polygonlist, diamond_poin
tlist, V(2,i), V(1,i  ), V(2,i+1), &DIAMOND_COLOR );
   75:                   addtriangle( diamond_polygonlist, diamond_poin
tlist, V(2,i), V(2,i+1), V(3,i  ), &DIAMOND_COLOR );
   76:                   addtriangle( diamond_polygonlist, diamond_poin
tlist, V(3,i), V(2,i+1), V(3,i+1), &DIAMOND_COLOR );
   77:                   addtriangle( diamond_polygonlist, diamond_poin
tlist, V(3,i), V(3,i+1), V(4,i  ), &DIAMOND_COLOR );
   78:           }
   79:           /* 上面……57面めだがSLASHは4角形までしか扱えないので8角形を3分割している
*/
   80:           addtetragon( diamond_polygonlist, diamond_pointlist, V
(0,3), V(0,2), V(0,1), V(0,0), &DIAMOND_COLOR );
   81:           addtetragon( diamond_polygonlist, diamond_pointlist, V
(0,7), V(0,4), V(0,3), V(0,0), &DIAMOND_COLOR );
   82:           addtetragon( diamond_polygonlist, diamond_pointlist, V
(0,7), V(0,6), V(0,5), V(0,4), &DIAMOND_COLOR );
   83:
```

▶「ストリートファイターIIダッシュ」はESCキーでポーズ。ポーズ中にHELPキーで画面モード変換です。　西條 勝(21)宮城県

```
 84:         AddNorm( diamond_polygonlist, diamond_pointlist );
 85:
 86:         return;
 87: }
 88:
 89: /* 形状を破棄する */
 90: void    destroyDiamond()
 91: {
 92:         free( diamond_polygonlist );
 93:         free( diamond_pointlist );
 94:         return;
 95: }
```

リスト10 jewel.c

```
  1: /*
  2:         jewel.c
  3:         - SLASHをCから利用する最低限のサンプルプログラム
  4:           May 1994       丹 明彦(Oh!X)
  5: */
  6:
  7: #define         __IOCS_INLINE__
  8: #include        <iocslib.h>
  9: #define         __DOS_INLINE__
 10: #include        <doslib.h>
 11: #include        <stdio.h>
 12: #include        <stdlib.h>
 13:
 14: #include        <slash2/slashlib.h>
 15: #include        <slash2/doublebuffer.h>
 16: #include        <slash2/timedifference.h>
 17: #include        <slash2/perfmon.h>
 18:
 19: #include        "diamond.h"
 20:
 21: SLTRANSWORK     *work;
 22: SLMINMAX        *minmax1, *minmax2, *minmaxt;
 23: SLPARAMETER     parameter;
 24:
 25: double  fps;                            /* 1秒間に何フレーム描画したか */
 26:
 27: #define NOBJECT         1               /* 表示する物体数 */
 28:
 29: void    main()
 30: {
 31:         int     sp;
 32:         int     t1, t2, dt;
 33:         int     angle = 0;
 34:         int     time = 1;
 35:
 36:         /* ワークエリアのための領域確保 */
 37:         work = malloc( sizeof(SLTRANSWORK)*(DIAMOND_NPOINT+1)
);
 38:         minmax1 = malloc( sizeof(SLMINMAX)*(NOBJECT+1) );
 39:         minmax2 = malloc( sizeof(SLMINMAX)*(NOBJECT+1) );
 40:
 41:         /* クリア用MINMAXワークの初期化: 初回は全画面クリアさせる */
 42:         minmax1[0].xmin = minmax2[0].xmin = 0;
 43:         minmax1[0].ymin = minmax2[0].ymin = 0;
 44:         minmax1[0].xmax = minmax2[0].xmax = 255;
 45:         minmax1[0].ymax = minmax2[0].ymax = 255;
 46:         minmax1[1].xmin = minmax2[1].xmin = -1;
 47:         minmax1[1].ymin = minmax2[1].ymin = -1;
 48:         minmax1[1].xmax = minmax2[1].xmax = -1;
 49:         minmax1[1].ymax = minmax2[1].ymax = -1;
 50:
 51:         /* ダイヤモンドの形状を作成する
 52:            diamond_polygonlist,diamond_pointlistの領域が確保される *
/
 53:         createDiamond();
 54:
 55:         /* 画面の初期化 */
 56:         /*CRTMOD( 14 );*/       /* 31kHz / 256×256ドット / 6553
6色モード */
 57:         CRTMOD( 15 );           /* 15kHz / 256×256ドット / 6553
6色モード */
 58:         G_CLR_ON();
 59:         B_CUROFF();
 60:
 61:         /* スーパバイザモード */
 62:         sp = SUPER( 0 );
 63:
 64:         /* グラフィックVRAM用描画関数の初期設定 */
 65:         SetClearColor( RGBILONG(0,0,0,0) );
 66:         SetWindowSize( 256, 256 );
 67:         SetWindowCenter( 128, 128 );
 68:         SetReverse( 2 );
 69:
 70:         /* パフォーマンスモニタをリセット */
 71:         PMreset;
 72:
 73:         /* 経過時間を計測する */
 74:         t1 = ONTIME();
 75:
 76:         /* ダブルバッファリングを開始する */
 77:         setDoubleBufferMode( 1, 0 );    /* グラフィックVRAMのみ */
 78:         startDoubleBuffer( 2 );         /* 最短2/60秒でページ切り替え
*/
 79:
 80:         /* メインループ */
 81:         for (;;) {
 82:                 /* ESCキーで終了する */
 83:                 if ( BITSNS(0x00)&2 ) {                 /* 押さ
れたら */
 84:                         while ( BITSNS(0x00)&2 );       /* 離さ
れるのを待って */
 85:                         break;                          /* ルー
プを脱出する */
 86:                 }
 87:
 88:                 /* これから描画するページが裏ページになるのを待つ */
 89:                 waitDoubleBufferSync;
 90:
 91:                 /* 描画ページを切り替え,画面をクリアする */
 92:                 if ( (time%2) == 1 ) {          /* 1,3,5,……回
目 */
 93:                         SetClearPlane( (unsigned short *)0xC00
000 );
 94:                         ClearBox( minmax1 );
 95:                         SetWritePlane( (unsigned short *)0xC00
000 );
 96:                         minmaxt = minmax1;
 97:                 } else {                        /* 2,4,6,……回
目 */
 98:                         SetClearPlane( (unsigned short *)0xC00
200 );
 99:                         ClearBox( minmax2 );
100:                         SetWritePlane( (unsigned short *)0xC00
200 );
101:                         minmaxt = minmax2;
102:                 }
103:
104:                 /* 前回の時刻との差 */
105:                 t2 = ONTIME();
106:                 dt = TIMEDIFFERENCE(t2,t1);
107:                 t1 = t2;
108:
109:                 /* 時間差をもとに回転角を加算する */
110:                 angle += dt*2;
111:                 angle %= 4096;
112:
113:                 /* 光源の方向(左手前) */
114:                 parameter.alpha = 0;
115:                 parameter.beta  = 8;
116:                 /* 通常のシェーディングを行う */
117:                 parameter.palettype = SLPALETTYPE_NORMAL;
118:                 /* 3軸の回転角を計算する */
119:                 /* それぞれ倍率が異なるのは見栄えよく回転させるため */
120:                 parameter.head  = (angle*5)%4096;
121:                 parameter.pitch = (angle*7)%4096;
122:                 parameter.bank  = (angle*9)%4096;
123:                 /* 平行移動なし(物体は画面中央で回転) */
124:                 parameter.x = 0;
125:                 parameter.y = 0;
126:                 parameter.z = 300;      /* 全体が画面に入る程度に奥に動
かす */
127:
128:                 /* 座標変換 */
129:                 TranslateAll( &parameter, work, diamond_pointl
ist, minmaxt );
130:                 /* 描画 */
131:                 DisplayPolygonList( diamond_polygonlist, work,
 minmaxt );
132:                 /* クリア用MINMAXワークエリアの調整 */
133:                 minmaxt = AdjustMinMax( minmaxt );
134:
135:                 /* パフォーマンスモニタのカウントは1周期に1回呼ぶ */
136:                 PMcount;
137:
138:                 /* 表示ページ切り替えを割り込みルーチンに許可する */
139:                 toggleDoubleBufferApage();
140:
141:                 /* ループ回数を加算する */
142:                 time++;
143:         }
144:
145:         /* ダブルバッファリング終了 */
146:         endDoubleBuffer();
147:
148:         /* パフォーマンスモニタから平均fps値を得る */
149:         fps = PMaverage;
150:
151:         /* グラフィック画面をクリア */
152:         SetClearPlane( (unsigned short *)0xC00000 );
153:         ClearBox( minmax1 );
154:         SetClearPlane( (unsigned short *)0xC00200 );
155:         ClearBox( minmax2 );
156:
157:         /* ユーザモードに戻す */
158:         SUPER( sp );
159:
160:         /* 画面を戻す */
161:         B_CURON();
162:         CRTMOD( 16 );
163:
164:         /* キーバッファをクリアする */
165:         KFLUSHIO( 0xFF );
166:
167:         /* ダイヤモンドの形状を破棄する */
168:         destroyDiamond();
169:
170:         /* ワークエリア用の領域を破棄する */
171:         free( minmax2 );
172:         free( minmax1 );
173:         free( work );
174:
175:         /* 計算した平均fps値を表示して終了する */
176:         printf( "average = %2.2ffps     ¥n", fps );
177:
178:         return;
179: }
```

SIDE B

SIDE B
テクスチャマッピングを考える

Yokouchi Takeshi 横内 威至

今回は，最近，特殊効果としてゲームの世界に定着しつつある
テクスチャマッピングのアルゴリズムを確認していく
そして，ただの自由変形ではない，正しいテクスチャマッピングの実現を目指す

「デイトナUSA」をやってみた。アンダーステアが厳しすぎる。もしかしたらリアルなのかもしれないが，ヴォーカルの「デーイートーナー」がやたら抜けているのでもういやだ。一発でやる気がなくなってしまった。ゲーム自体は気にいらなくても，車に映り込む雲はかなり気持ちいい。それ以外ではやはり「リッジレーサー」のムードのほうが俺は好きだ。もうひとつ個人的な趣味であるが，セガのグラフィックはどうも好きになれないのだ。

でも，とりあえずテクスチャマッピングは偉大だよな。セガもナムコもこれ以降疑似3Dモノから，本格的な3Dモノへ移行していくのではないだろうか。セガの場合，いままでの疑似3Dモノのムードがそのまま表れてるし，そのまますんなり移行してくるのでは？　ところで，いまさらながらバーチャレーシングにはまってしまった。特に20周バージョンがとてつもなく熱い。以前はなんのためにピットがあるのかわからなかったが，ようやく理解できた。グリップが回復したら一気にタイムアタック。現在上級48秒65が自己ベスト。中級はとても攻められないからパス。バトルも熱い。1秒差を積み重ねる快感。スリップストリームで少しでも稼いで捲る。バトルになると20周でも足りない。せめて，80周ぐらいはやりたい(設定はできるらしい)，いつかはルマン24時間耐久もやってみたいと思う次第である。

ということで今月は突然だがテクスチャマッピングを考えようと思う。なぜいきなりこのテーマを取りあげることにしたかというと，マッピングを使えば，モデラの各座標軸で構成される平面に立体的な3面図を張ることもできるので，かなり理解しやすくなるのでは，と考えているからである。当然，律儀な方法で実現しようとは思っていない。用途を考えるとそこそこ高速でなければ困る。いつものように姑息な技を多用してコーディングしようと思う。どこまでを整数演算でカバーできるか，どの程度の誤差まで許されるか，さらに解像度が低くてもかまわないだろうか。とりあえずやってみよう。駄目だったらまた別の方法があるはずだ。

エセマッピングについて

まずマッピングの実例として写真1と写真2を見てもらおう。元絵に適当なものがなかったので，手元にあった写真を取り込んで使った。写真1は「取り込んだ画像をマッピングしてみました」といえば皆納得するだろう（少々の歪み，誤差は許してもらいたい。たいしたプログラムではないので，細かいところでいくらでもボロが出ている)。では写真2で

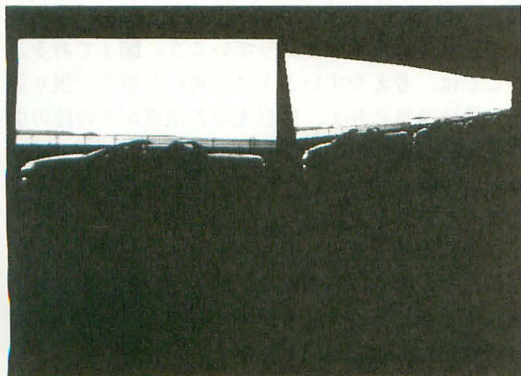



写真1：本物に近いマッピング

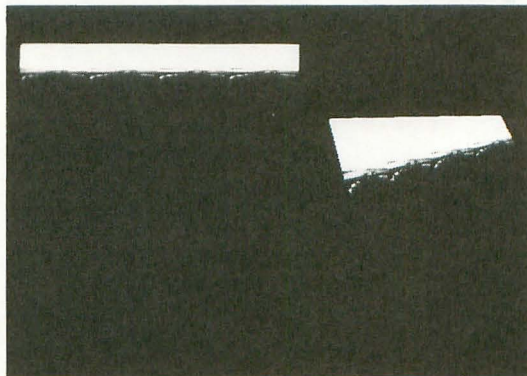



写真2：ウソのマッピング

ある。この写真を見せられて「マッピングしています」といわれても，あからさまにウソだとわかる。結局のところ写真1が正確なマッピング，写真2がエセマッピングなのである。ひと目でわからない場合は，図1と照らし合わせれば，どちらがより正しくマッピングされたものか見抜けるであろう。

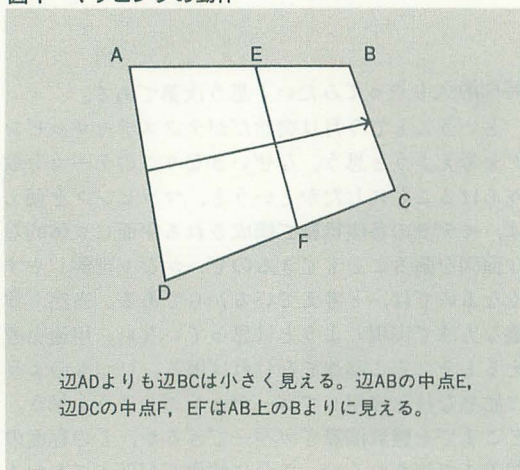
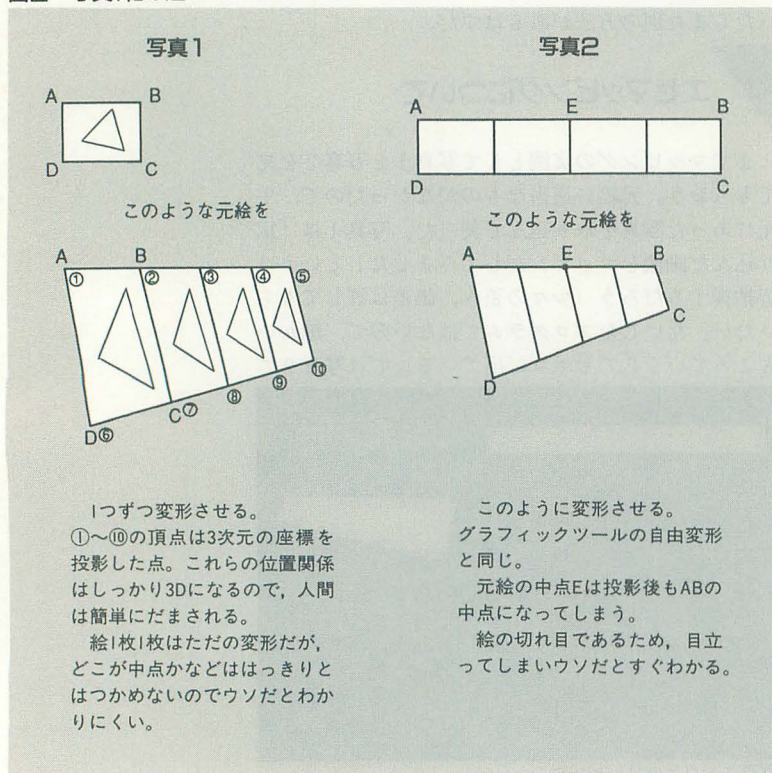
さて，簡単に正解をばらしてしまったのだが，解説を読まず，写真を見比べただけで写真2がウソだと気づいた人はいたであろうか。実は写真1，2ともに同じアルゴリズムで作られた画像なのである。どうして同じアルゴリズムで違いが出るかは，図2

図1　マッピングの動作

辺ADよりも辺BCは小さく見える。辺ABの中点E，辺DCの中点F，EFはAB上のBよりに見える。

図2　写真1,2の違い

1つずつ変形させる。
①～⑩の頂点は3次元の座標を投影した点。これらの位置関係はしっかり3Dになるので，人間は簡単にだまされる。
絵1枚1枚はただの変形だが，どこが中点かなどははっきりとはつかめないのでウソだとわかりにくい。

このように変形させる。
グラフィックツールの自由変形と同じ。
元絵の中点Eは投影後もABの中点になってしまう。
絵の切れ目であるため，目立ってしまいウソだとすぐわかる。

を見てもらいたい。写真2は4つに分割された各平面に写真を張っているだけなのに対し，写真1は，まがりなりにも3D計算をしてその分割点を求めているため，より正しくマッピングされたと人間の目には映るのだ。

よって，1枚単位で見ると実は写真1もウソをついているのである。人間の目なんてのはこんなもんであろう。ただし，これが動画像となるとやや異様な感じを受けるかもしれない。

このエセマッピングは非常に簡単なアルゴリズムで実現できる。といっても清く正しいマッピングに比べれば非常に簡単なだけであって，面倒なことには変わりはない。まあ，ただの自由変形である，というだけでわかるであろう。簡単なアルゴリズムは図3に解説しておいた。基本的にはブレゼンハムのラインルーチンを複雑に絡めた形となっている。

本物のマッピングについて

それでは本物のマッピングについて考えてみたい。エセマッピングとの差は微妙だが明らかに違うものである。攻略としては，やはり自由変形のように考えてみたい。どこで差ができるかといえば，エセマッピングがブレゼンハムルーチンをそのまま使っていた部分，つまりは辺を得るときなどに1次関数を使っていた部分がポイントである。この1次関数は，1ドットごとの差が等しい等差数列を使った。

しかし，実際のマッピングでは，

$$P=p/z$$

の式で投影されると考えると，等比数列的に画像が張られねばならないはずである。その比率をどのように計算するか，どのように処理するかがマッピングの鍵となっており，現在ではどの程度まで誤魔化せるかはわからない。つまり，単純に自由変形した画像を張りつけるだけでは，マッピングとはいえないのだ。

アルゴリズムを練る

では具体的な手順を追っていこう。図4である。ここでは，考えやすいように三角形を扱う。張り込む元絵は適当に与え，変形先の各頂点がその絵の中のどこに対応しているかを任意に設定できることにする。一応，完成するとかなりフレキシブルなモノになるであろう。

さて，漠然と考えるとどこから入っていっていいのかわかりづらい。まずはどのように画面上をスキャンするかである。速度を考えると当然，無駄な点を計算したくはない。そうなると，普通に垂直にスキャンしていくのが速そうに思える。

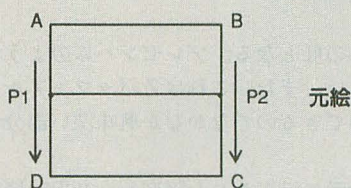
律儀にやるならばいろいろな方法が考えられる。グラフィックシステムなどに使うのであれば，速度よりも画像が優先されるので各ピクセル単位で色の補間をしたりすることも考えなくてはならない。たとえば元絵のすべてのドットに対して座標変換を行い，点が重なっていれば色を合成したりすればいい。

しかし，今回はそこまでの精度を要求しないだろうし，そこそこの高速化が必要なので普通のポリゴンと同じように垂直にスキャンすることにする。

全体の流れを簡単に考えると，

1) まずは各辺をＹ方向にスキャンしていき，Ｙ方向１ドットごとのＺ座標を得る
2) 水平１ラインごとに描画。先に得た左右の辺のＺ座標によって元絵をうまく張っていく
3) Ｙが最下段になるまでループ

以上のようになるだろう。かなり大雑把な手順だが，主な処理はこんなところであろう。もちろん，1)の「Ｙ方向１ドットごと」は投影された画面上でのこと。図５のように各Ｚ座標を求める。これが高速化，手抜きの肝心な部分。もしまともなマッピングを行うならば1)，2)の過程でドット単位の補間が必要。各ドットに，元絵のどの領域がどの程度入っているかで色を修正しなければならない。

図3　自由変形ルーチン

ポインタ1(P1)はAD上を，ポインタ2(P2)はBC上を移動。それぞれブレゼンハムのルーチンに従ってAD，BC上に置かれる。同じタイミングでP1はDに，P2はCになるよう，カウンタをうまく合わせておく。

あとはP1，P2が元絵のどこにあたるかを調べてその部分を描画する。当然ブレゼンハムのルーチンでOK。

図4　マッピングの方法

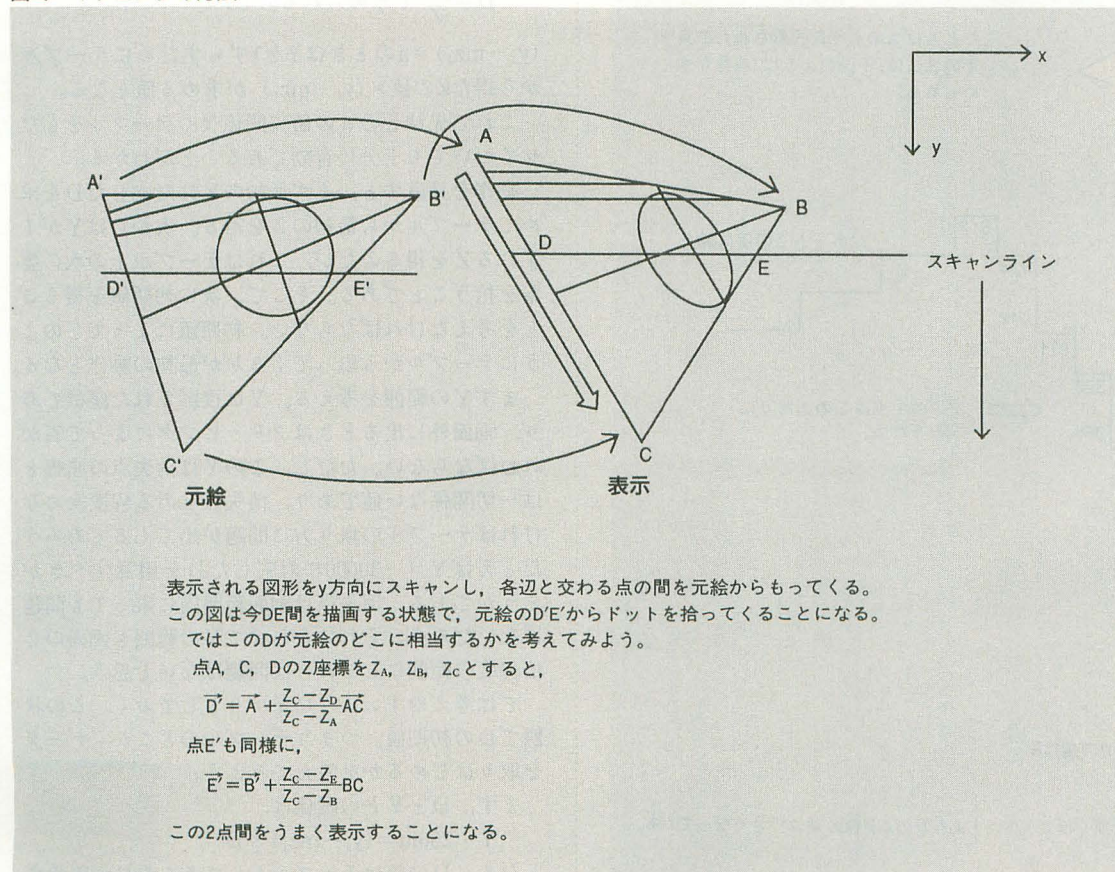
表示される図形をy方向にスキャンし，各辺と交わる点の間を元絵からもってくる。
この図は今DE間を描画する状態で，元絵のD'E'からドットを拾ってくることになる。
ではこのDが元絵のどこに相当するかを考えてみよう。
点A，C，DのZ座標をZ_A，Z_B，Z_Cとすると，

$$\overrightarrow{D'}=\overrightarrow{A'}+\frac{Z_C-Z_D}{Z_C-Z_A}\overrightarrow{AC}$$

点E'も同様に，

$$\overrightarrow{E'}=\overrightarrow{B'}+\frac{Z_C-Z_E}{Z_C-Z_B}BC$$

この2点間をうまく表示することになる。

ハードコア3Dエクスタシー(第10回)

エッジ検出

これが全体の肝となる。ブレゼンハムのように単純にはいかない。また，これはZバッファアルゴリズムにも応用できるのでなかなか興味深い部分でもある。

問題は2次元に投影された図形から3次元情報を得ることである。まず，2次元に投影するときの方法を思い出してみよう。2点(x,y1,z1)(x,y2,z2)によって作られる直線の投影を考えてみる。SLASHの仕様に合わせて投影すると，この線上の点(x,y,z)は画面上で(X,Y)に投影されるとすると図6のように考えられる。

ここで求められるYが，画面に投影されたときの画面上での座標となる。要するに，このYを上から下まで考え，それぞれに対応したZ座標を図5のように求めることになる。次に図7で考えてみよう。式を見ると，このYを決定する変数のうちm,(y_1-mz_1)は定数となる。よって，このYを増減させる唯一の変数はzということになる。だが，図7のようにYを1ずつずらすためには，かなり労力を要することになる。

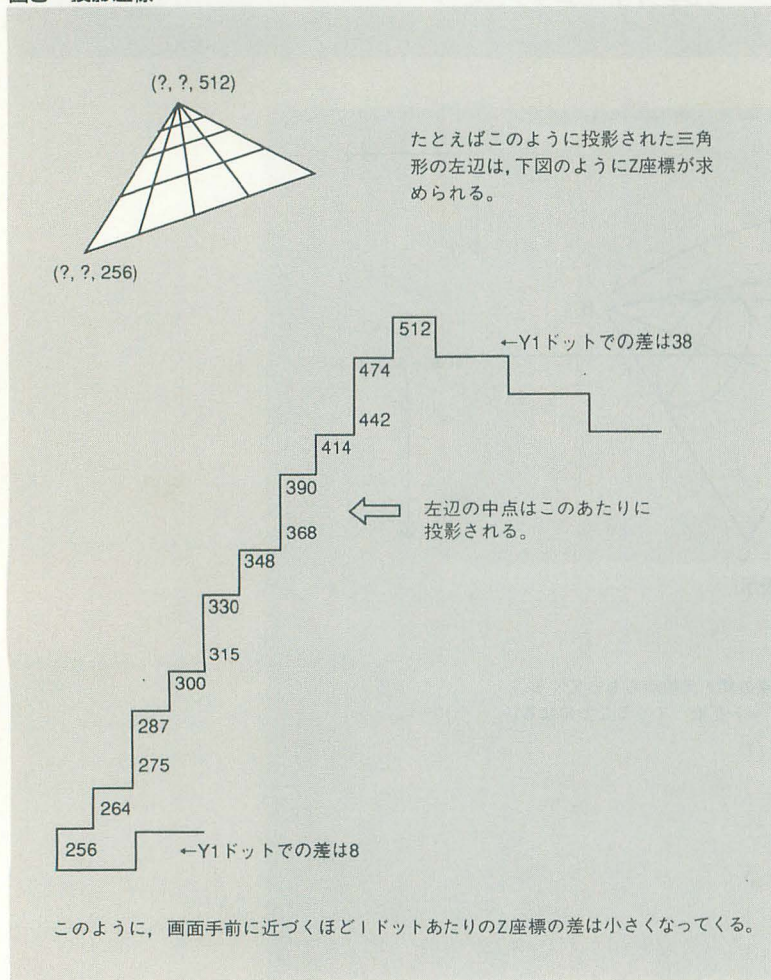



図5　投影座標

このように，画面手前に近づくほど1ドットあたりのZ座標の差は小さくなってくる。

一般論をはずす

そろそろ現実的な問題も含めながら考えていこう。まず，前の部分で問題となったzの絡み方，図7のa/zはテーブルにすることでがんばろうと思う。aの値によって困ったことが起こると考えられるが，テーブルにする値は，

D=256×$10000/z

としておく。$10000は小数値を残すための下駄はかせである。そして，このDがなるべく整数値をとるようなzをあらかじめテーブルとしてもっておけば，割り算を削ることができる。

$$Y=256\left(\frac{y_1-mz_1}{Z}+m\right) \text{ より，}$$

$$Y-256m=256\frac{y_1-mz_1}{Z}$$

$$=(y_1-mz_1)\ D \text{ とおく}$$

テーブルには $(y_1-mz_1)=1$の状態でのD (1, 2, 3, ……) に対応したZが入っている。

$D=\frac{256}{Z}$で求められるので，

$(y_1-mz_1)=a$のときはYを1ずらすためにテーブルから得たZの値×(y_1-mz_1) が求める値となる。

これで先ほどのaの値に関係なくテーブルをもたせておいても十分に有効であることがわかる。

動作を見直すと，まず最初のYに対応したDを求めてテーブルから最初のZを得る。次からはYが1ずれるZを得るのだが，これはテーブル上の次の要素を拾うことである。そこで，次に初期値を得ることを考えなければならない。初期値によってどのようにテーブルから取ってくるかが最初の動作となる。

まずYの範囲を考える。Yは投影された座標であり，画面外に出るときはクリッピングによって省かれねばならない。ただし，このYは消失点の座標とは一切関係ない値であり，消失点をある程度決めなければテーブルの取り方に問題が出てくるであろう。たとえばY=−30000に対応したDを用意すべきかどうかである。消失点は画面範囲内に限っても問題はないだろう。それに合わせてYの範囲も画面の2倍程度の範囲にしておけば問題はないと思う。

では考えやすいようにY=0としておく。この状態でDの初期値，つまりテーブルのどこからデータを取りはじめるかを考えてみよう。

まず，DとYとの関係は，

$$Y-256m=(y_1-mz_1)\times D$$

となる。Dの値はテーブルとしてあらかじめ用意す

る。配列のようにD(n)として表すと，D(1)～D(3)には，それに対応したZが入っている。ただし，$(y_1-m_1)=1$としたときの値である。そうでないときは，Zの値に(y_1-mz_1)をかけた値が求める値となる。よって，(y_1-mz_1)を無視。Y=0とすると，

$$D=-256m$$

となる。すると，この段階からテーブルを読み始めることになる。

では，y_1の値はどうなるか。Yは画面上の座標であるから，中央を0として，$-128\leqq Y\leqq 128$程度に収まる。気にするような大きさではないので，mをもとにテーブル化するのが正しいであろう。

精度を考える

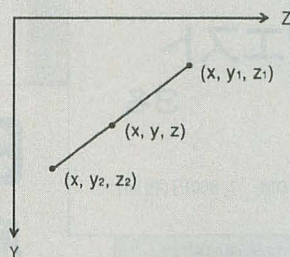
ということでマッピングもかなり現実的なものとなってきたが，追求すべきは高速性。実数演算を使うのは避けたい。だからといって，固定小数点演算では厳しい部分が目立ってくる。どのあたりで計算がオーバーするかといえば，直線の傾きmが絡んでくる部分である。少し考えてみてほしい。つまりは，mというのは，一般的に考えると無限大になってしまう数値だからである。完全にmの分子，z_1-z_2が0のときは単純な拡大縮小でカバーできるので，それ以外の状況を考えてみないとならない。

あと，座標系を－32768～＋32767とすると，mの最大値は65535である。ワードで収まる範囲ではあるが，符号も考慮しなければならない。符号だけ別で考えればなんとかなるのだが，途中で鍵となっていた数式，(y_1-mz_1)も単純には収まらない値となる。そのような状況ではほとんどマッピングをしても意味がなさそうである。どちらにせよ，メモリの都合もあって元絵自体大きいものにはならないので，そこまでの範囲をカバーする必要はなさそうだ。範囲を抑えることでこの値もロングワードで収められるであろう。実際にコーディングしていくうえでこれらの値を調整することにしよう。どうせほかの問題も絡んでくるのだろうから，ある程度見切ったら即実行，が望ましい。

続く

あとはこれらの調整。ここまで考えればなんとかなりそうな感じがしてきたであろう。リストが掲載できる範囲になるかどうかはまだわからないが，アルゴリズムだけは固めておきたいと思う。おそらくリアルタイムでできるほど甘いものではなさそうだが，それならそれで偽マッピングぐらいはシステムに載せることになるかもしれない。単純な自由変形だけでも案外識別できないものだ。注意して見ていれば気づくだろうが，全体が動いていればかなり騙すことができる。少なくとも私は騙される。

せっかくだから，しっかりとした誤差を突き詰めて色の補間も行うマッピングも作りたいと思うが，やっぱりやらないだろう。ほかにすべきことがまだまだある。再試験なんかもかなりのプレッシャーとなっている。ということでまた来月。

図6　投影面の計算式

YZ平面から見ると左図のようになる。
これより，傾きmは，

$$m=\frac{(y_2-y_1)}{(z_2-z_1)}$$

これを用い，この直線上の点は

$$y=y_1+m\ (z-z_1)$$

さて，これを画面に投影すると，SLASHの仕様に合わせて，

$$Y=\frac{y}{z}\times 256$$

先のyを代入して，

$$Y=256\left(\frac{y_1}{z}+m-\frac{mz_1}{z}\right)$$

$$=256\left(\frac{y_1-mz_1}{z}+m\right)$$

Xについても同様だが，ここで重要なのはYに対応したZ座標である。

図7　Yの値を求める

$$Y(z)=256\left(\frac{y_1-mz_1}{z}+m\right)$$

このYがY(z_1)からY(z_2)の間の整数値をとる。Y(z_1)は実際は整数にならないかもしれないが，座標値なので整数にまるめる。Y(z_2)も同じ。

では，どうすればY(z)は1ずつ増減させられるであろうか。

$$Y(z)=256m+\frac{a}{z}\qquad (a=256(y_1-mz_1))$$

これより，$\frac{a}{z}$が1変わるようにzを見つければよい。

当然，グラフはこのようになる。変位1になるように
Zを定めていくのは容易ではない。

愛読者プレゼント

1 スーパーリアル麻雀PIV

1名

X68000用

5″2HD版　12,800円(税別)

▲ビング　☎03(3492)1079

7月号で紹介した麻雀ゲーム3種のプレゼントです。どれにしようか迷っちゃうかな?

2 雀神クエスト

3名

X68000用

5″2HD版　7,800円(税別)

▲SPS　☎0245(45)5777

3 麻雀航海記

3名

X68000用

5″2HD版　5,800円(税込)

▲TAKERU　☎052(824)2493

4 「ツインビーPARADISE」下敷き

非売品

「(善)のゲームミュージックでバビンチョ」でも紹介したCDの発売記念グッズです。絵柄は2種類。どちらか選んでね。

A　3名　「ツインビーPARADISE－MUSIC BOX－」

色はピンク

B　3名

「ツインビーPARADISE/熱唱!ボーカルバトル編」

色は青

▲キングレコード　☎03(3945)2122

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ,希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1994年8月18日の到着分までとします。当選者の発表は1994年10月号で行います。雑誌公正競争規約の定めにより,当選された方はこの号のほかの懸賞には当選できない場合がありますので,ご了承ください。

6月号プレゼント当選者

1Easydraw SX-68K　(埼玉県)加茂択也　桐原直義　(長野県)吉田昌彦　(兵庫県)芝田　稔　(大分県)黒木千尋　2Communication SX-68K　(東京都)西村雅文　(静岡県)鈴木典雄　坂田清明　堀口明彦　(大阪府)西嶌郁夫　3SX-WINDOWデスクアクセサリ集　(千葉県)栗本興一　(東京都)高山彰広　(神奈川県)片平正二　鈴木幹雄　(鹿児島県)手嶋和徹　4CYBERNOTE PRO-68K　(神奈川県)鈴木幸太　萩原保憲　(福井県)山本直行　(徳島県)武林俊宏　(沖縄県)伊舎堂盛行　5コットン　(東京都)間島恒己　石井大輔　(広島県)池岡　慎　6テレホンカード　(千葉県)白井宏尚　ほか9名　7ネメシス'90改　(埼玉県)吉田佳子　飯島寿治　布施泰雄　8ロボットコンストラクションR.C.　(北海道)按田一歩　(東京都)森脇弘之　9ロボット集+α　Vol.2　(東京都)山本一朗　(兵庫県)秋山祐司　(広島県)中光乙孔　10ファイナルファイト　(広島県)周田満雄　11ストリートファイターIIダッシュ　(佐賀県)村上淳一　12ヨーロッパ戦線　(大阪府)森山隆一　13ロイヤルブラッド　(宮城県)小松明博　14リーディングカンパニー　(千葉県)石田晴幸　15太閤立志伝ハンドブック　(神奈川県)加藤　修　(和歌山県)清水良輔　16三國志IIIハンドブック　(千葉県)笹原修達　(長野県)坂下　努　17伊忍道ハンドブック　(東京都)塩野浩之　(三重県)松野秀人　18エキサイティングアワー/出世大相撲　(大阪府)大久保貴司　(兵庫県)宮本智幸　橋本　誠　19アルゴスの戦士　(埼玉県)大塚俊宏　(千葉県)菊井一在　(徳島県)瀬良信一　20ポスター　(栃木県)尾形　靖　(群馬県)井上博健　(神奈川県)佐々木利男　今井礼治　(宮崎県)山口宏幸　21ポストカード　(大分県)穴井浩二　ほか9名　22テレホンカード　(北海道)藤田真史　(埼玉県)松永龍次　(和歌山県)本田正彦　(富山県)折谷祐一　(岡山県)定兼　丘　23宝魔ハンターライム　(群馬県)周東正男　(茨城県)秋山真一　(神奈川県)藤巻　太　24餓狼伝説　(愛知県)市川博基　神野力　(鹿児島県)蔵原洋一郎　25餓狼伝説2　(広島県)田中由美　(福岡県)中村志郎　26Free Software Book　(青森県)前田桂史　(埼玉県)塚原　一　(神奈川県)市川　功　(山梨県)藤沢吉文　(京都府)吉田豊廣　27040turboステッカー　(北海道)三上誠一　ほか19名　28X1/X1turbo用福袋　(北海道)野村明弘　(埼玉県)野口将人　(東京都)北浦暁光　(神奈川県)高橋伸之　(山梨県)高野真樹　29MZ-2500用福袋　(徳島県)五藤章博　(福岡県)古川　泉　30シャープペンシル　(大阪府)渡辺久孝　ほか9名

(敬称略)

以上の方々が当選しました。商品は順次発送いたしますが,入荷状況などにより遅れる場合もあります。

アナーキテクト宣言

AppleIIはパンクだった

最近は「パンクな」マシンを見かけなくなってきました。どのパソコンもみんなずいぶんとお行儀がよくなって信頼できるようになってきました。要するに，ビジネスでも十分に使えるようになってしまったのです，残念ながら。

ビジネスで使えて稼げる優等生というイメージのマシン，それと対極にあるパンクなイメージのあるマシンというものがあるとするならば，それは僕にとってはMacintoshが出る前のあのAppleIIです。お金をきちんと稼ぐことができるどころか，次の瞬間になにをするか予測もできない荒くれ者，それがAppleIIでした。

ソフトウェア的にアクセス制御されているディスクがクイックイッ，ガーガーとなっては，あっと驚くようなプログラムが走り出しました。背中に拡張カードを挿し込んでいくたびに，まるで別のマシンのように変身してくれました。何枚も挿したカードが，接触不良で動かなくなっても，「エイヤッ，動け！」とぐいっと上から両手で押すと，だいたいクイックイッ，ガーガーといいながら動き出したものです。

色がぼやっとにじんでいた画面も独特な魅力でした。画像鮮明なフルカラーより，ぼやっとにじむ何色かのほうが実際格段によかったのです。みんなで目をどろどろにしながら徹夜で遊んだものでした。

どのソフトも芸術的でした。作った人のアート心がダイレクトに伝わってきました。プロジェクトを組んで分業しながら完成させたものなど芸術ではありませんよね。

正確にいえば，AppleIIというのは不適当かもしれません。純正のAppleIIは当時は東レが代理店として扱っていまして，確か70万円か80万円もする代物でした。大学の研究室にはその純正版がおいてありましたが，個人で買うのはモノ好きで，しかもよっぽどの金持ちだけだったのです。

僕らが持っていたのは正式名称(？)「ニセApple」です。基板，これは確か香港か台湾製だったと思いますが，これを買い，次に部品を買います。AppleII用の部品がセットになって売られていました。最後にケースを買えば準備はOKです。ケースもものによって微妙に異なっており，これは純正に近い色をしているとかいいながら買ったものです。

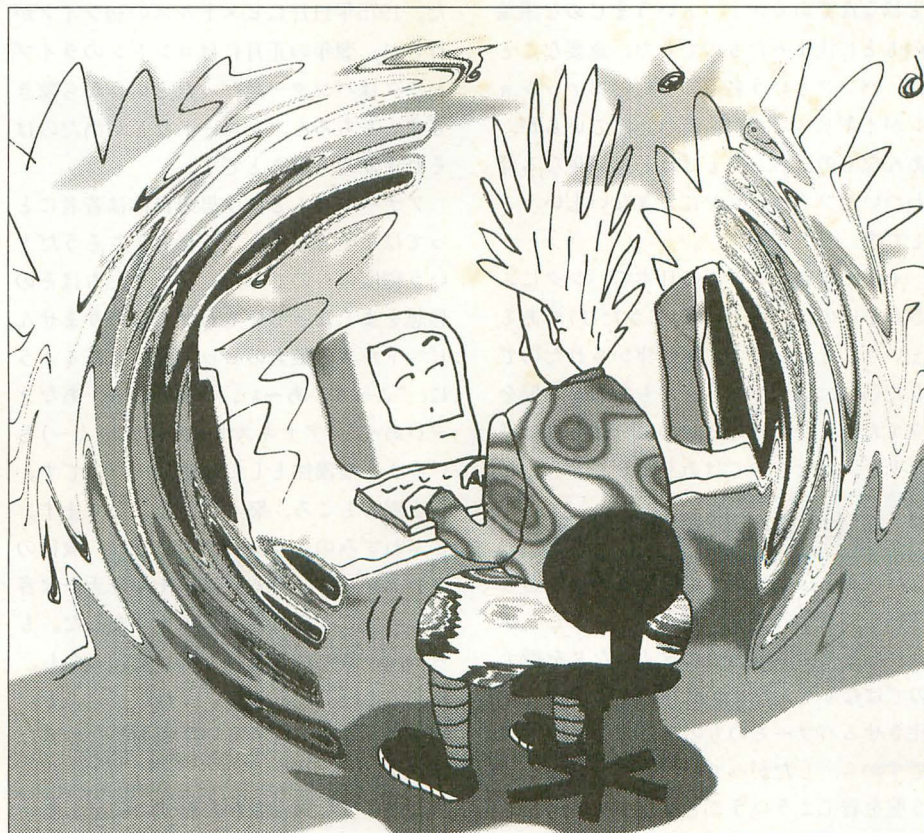
そして，あとは部品を1個1個ハンダ付けしていけばよいわけですが，そう簡単でもありません。特に画面出力の信号を作るところなどはアナログですので，微妙な調整が必要でした。僕はかなり苦しんで完成させましたが，先輩のおかげで十数万の出費をパーにしないですみました。

もともと，スティーブという同じ名前をもつ2人のヒッピー野郎がガレージカンパニーを作って売り始めたAppleIIのそのまた海賊版ですから，本当に怪しげなパンクなマシンでした。しかし，僕たちを魅了しまくってくれたのです。

AppleIIのその魅力が薄れてきたのは，現在のExcelやLotus1-2-3の先駆けであるVisicalcがAppleIIで普及してしまったころからでしょうか？　確かに「これは商売に使える」ということをわかりやすく教えてくれるソフトでした。

パンク野郎ここにあり

ずっと隠していましたが，もうばらしましょう。僕はパンク野郎です，実は。パンクとはなにかということはおいおい考えることにして，僕が正真正銘のパンクであるという例をとりあえず2つほど示すことにしましょう。

ひとつ目の例は結婚パーティです。おえらいさんを招待して話を聞かされたり，堅苦しい披露宴をやるのが嫌なので，式は外国で当人たちだけで行いました。ここまではよかったのですが，やはり，友人知人を

illustration：Haruhisa Yamada

アナーキテクト宣言

集めてなにかはやらなくてはなりません。ということで，後日パンクバンドを呼んでただただ演奏してもらったのでした。

これがまた実に不評で，「もっと静かに話したかった……」という不平不満でブーブーでした。もっとも，何割かの人は喜んで酒を飲みながら踊り狂ってくれましたが。

2つ目の例はパンクの精神にも触れる，とてもよい話です。僕は，いうまでもなくパンクバンドであるセックスピストルズやクラッシュなどに狂っていました。ピストルズ解散後にジョン・ライドン（自殺したのはただのチンピラのシド・ビシャス）がパブリックイメージリミテッド（PIL）というバンドを結成しまして（初期のものは，ピストルズ時代よりもはるかに素晴らしい音楽でした），そのPILが来日するというので，僕らパンクスはとんでもない格好をしてコンサートに行ったのでした。

僕のそのときの格好というのが，パンクとはなんであるか？　というまじめな議論のもとに決まったものでした。重要なことは，パンクという名前のついたファッションがイギリスなどから流れ込んでいるが，あんなものはパンクとはほど遠い単なる浮わついたファッションにすぎないということです。

そして結局，いまこの日本でパンクにふさわしいのは，地下足袋（じかたび）であるということになって，僕が穿かされたのでした（喜んでいたが……）。もちろん，髪を立てたりとか，そのほかの部分はそれほどオリジナルなものではありませんでしたが。

にせパンクに見せつけろ!

パンクとはなにかと定義するのはもともと難しい話なのです。パンクということばは，あるモノや事柄やスタイルなどを指すのではなくて，なにか既存のものを壊し変化させるパワーそのものを意味しているのですから。したがって，パンクとはこういう服を着てこういうことをして……，とことばで定義できるようになってしまった瞬間にそれはパンクの説明になっているどころか，もはや，その説明によって定義されているような概念そのものがパンクの攻撃の対象になってしまうのです。

イギリスで発生したパンクは比較的わかりやすいものでした。暗い世の中で失業した若者の怒り，既存の体制への反発が，パンクバンドという音楽や反抗的なファッションになりました。その担い手や取り巻きたちをパンクと呼んだわけです。

とにかくお先真っ暗で行き詰まってしまった若者たちが現状打破のために行うパフォーマンス，激しい攻撃のことば，一本調子のリズム，たてノリ系の踊り，そして，つば，へど，ヤク，血などが渾然となって，壮絶な空間，そしてムーブメントが成立したのでした。

主に政治的な怒りでしたので，無政府主義思想（アナーキズム）ときわめてよくマッチングして，ムーブメントは加速されました。1975年11月にピストルズの初ライブが行われ，翌年の正月にはロンドンのライブハウスはパンク一色だったというから驚きです。もちろん，レコードなどが出たのはそれよりあとのことです。

アナーキズムという思想自体は若者にとっては手頃でなんとなく格好よさそうだという程度のものでした。パンクたちはその思想をよく知っているわけではありません。ピストルズの最大のヒット曲にもあるように，“こ～ず，あ～い，うぉなび～，あなき～ぃあ～”（アナキストになりたいよ～）というような漠然とした願望だったのです。

実際のところ，怒りは商売になります。どぶねずみのような若者の怒りは，流行のファッション，あるいは，キャッチーな音楽ということで世界中に広まりました。もちろん，日本にも上陸しました。しかし，そこに上陸したものは，「これがパンクというファッションだ」という表面的な形式だけです。

このような固定された形式に満足している「にせパンク」どもに対しても僕ら真のパンクの怒りは向けられます。そして，これこそが我々日本人としてのいまのパンクだということで，コンサートに集まった「にせパンク」どもに見せつけるために，恥ずかしいのをがまんして地下足袋を穿いて行ったのです。

そうはいうものの，パンクなんてしょせん，子供がいやだいやだといっているのとなにも違わないのではないか？　といわれれば，まあそのとおりでしょうということです。きわめて，基本的でしかも普遍的な感情の発露がベースとなっています。

しかし，子供の単なる反抗というだけでは，それぞれの人の個人的な怒りという枠に留まるのですが，なんらかの方向性が具体的にその怒りに合わさると初めてそこに大きな流れ，大きなムーブメントが作られ，結果として，社会のいろいろなところにその影響は染み渡っていくのです。

パンクはどこに行く?

最近，僕はいわゆるパンクファッションやパンクミュージックにのめりこむようなことがなくなってきました。なぜかといえば，明らかでしょう。僕の定義によれば，20年近く前に起こったパンクという形式やファッションが，現在もまだパンクであるわけがないのです。

そして今，なにをやっているかというと，計算機アーキテクチャだとか，人工生命などの研究をやっています。大きな変貌をとげたのでしょうか？　いいえいいえ，まだまだ僕は自分のことをパンクだと実は思っているのです。

パンクはなにか現状を打ち破る革新的なものと結びついて初めてそのエネルギーが満ち満ちてきます。その新しいなにかが僕が目指しているほうにあるのではないかと考えているのです。

パンク，カウンタカルチャ（反文化）
→　計算機関連の領域

という流れは確実にあると思っています。人の流れもそうですし，思想的な流れも少

なからずあるでしょう。この流れが明確になってきたのは，1980年前後です。

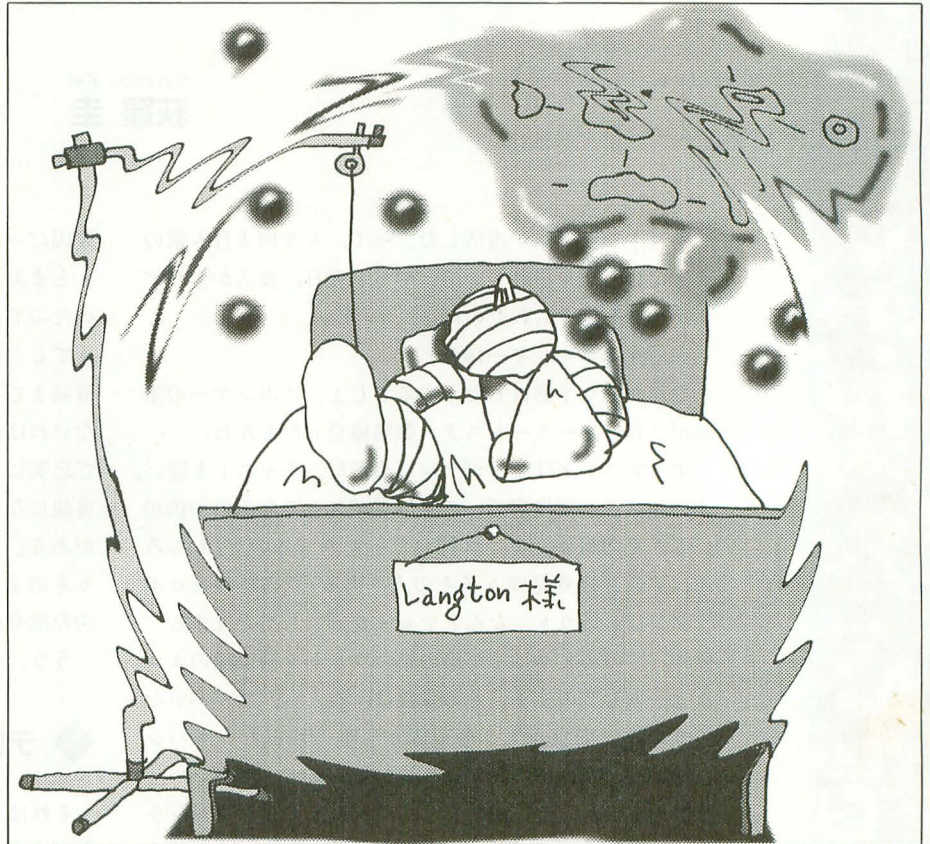
ちょうど計算機の領域でも，中央に1台メインフレームコンピュータが鎮座して，そのまわりに端末が並ぶという中央集権的な形態から，ワークステーション群が分散してつながれるというアナーキーな形態に移り始めた時期と符合します。この流れにおいて，AppleIIの存在が決定的な意味をもったことは疑いのない事実でしょう。

パンクというムーブメントがそこにおいてなにと結びついてきたのかということをもう少しはっきりいいましょう。要するに，次のような方向性をもつ道です。これが我々の前にさーっと開けてきたというわけです。まず，最も目の前に見えているのが，いわゆる双方向マルチメディア通信(マイクロソフトのドラゴンとか)であり，そのずっと先には人工生命や知能機械があり，さらにずっと先にはいわゆるサイバーパンクのような，いまはまだSFの物語としてしかとらえられていない領域があるといいたいのです。

この道はきわめて革新的なものですから，無数の障害物が大きくたちはだかっています。それをパンクのエネルギーは鮮やかに，そして過激にとっぱらっていってくれるのではないかと思われるのです。

35本骨折したときの夢

人工生命なんてのはずいぶんとパンクだと思います。たまたま現在の形をとっている人間なんてのは横に置いておき，もっと本質的なところにある「情報そのもの」を相手にしようというのが基本的な立場なのですから。情報＝生命ですよ。どうなんでしょう，海千山千の世界のように思われるのではないでしょうか？

このあいだ本連載でも紹介した，僕がやっているランギーモデルもとりようによってはずいぶんと過激な話です。生命とか知能とかのほとんど氷山の一角もわかっていないうちから，言語が発生するだの，これが方言だなどといっているのですから。

しかし，なにかが人工生命ムーブメントのなかで起こっていると確信しているのか，単なる興味半分かはわかりませんが，いろいろな人々が注目してくれています。ランギーモデルは，日経産業新聞の5月10日付の「先端技術」という欄で5段にわたって紹介されました。

「生物進化過程の言語発生を模擬」

とバーーンと紹介されているのです。記事の内容的には，「あー，記者さんはあまり理解してくれなかったなー」という感じですが，まあ雰囲気はそこそこ伝わっていると思います。

パンクにはジョン・ライドンがいましたが，人工生命にも教祖はいます。それがLangtonです。人工生命に関するワークショップを1987年に開いて今日の流れを作った彼もなかなかのパンクのように思えます。

ロックバンドを作りたかった彼はベトナム戦争に行くのを拒否して，病院で死体運搬の仕事をしていました。あるとき，死体がいきなり立ち上がり絶叫しながら走り出したのを見てその仕事をやめたなどと冗談半分に語っています。

また，ハンググライダーで落下し35本もの骨を折り肺に穴を開け顔をつぶして，入院中の半分意識のないときに，生命のパターンのようなものを見て，それから人工生命に対する思いが固まっただの，いかにも「教祖さまーっ！」という感じです。

不況の昨今，まるで浮き世離れした人工生命の研究ができるのは実にありがたいと思って，毎日小汚い，ほとんど「監獄」のように汚い(これは本当にここを訪れたお客さんのことば)建物の中をゆっくりと歩いています(廊下が狭すぎて角で人とぶつかるので)。

「いやー，それにしても，パンクでもそれなりに生きていけるのだな(実感)！」

石の言葉，言葉の夢

［第5回］

個人がメディアになる日

Ogikubo Kei
荻窪 圭

ウルトラセブンが復活した。って，もう何カ月も前のことだけど，見損ねていて，やっと先日，友人からビデオテープを借りられたのだ。

「太陽エネルギー作戦」

なんかやな予感はした。するでしょ。スポンサーの筆頭が「(社)ソーラーシステム振興協会」だもんね。

オープニングはオリジナルに忠実だ。キャストも凄い。オールスター総出演で，モロボシダンって名前の子供がいて，その母親がアンヌ隊員だったりするのだ(もちろん，アンヌは菱見ゆり子その人である)。当時のモロボシダンはというと，なんとナレーションしてたりする。傑作。このあたりはまだ序の口。ウルトラ警備隊のユニフォームも昔のまま。その隊長は，なんと，フルハシなのである。旧ウルトラ警備隊から唯一残ったフルハシ(毒蝮三太夫ね)が隊長になっている，ってとこがいい。腹は出てるが，渋い配役をしてくれたものだ。ウルトラホークも健在で，ポインターはさすがに現代の車を改造して使っているが，渋いカラーコーディネートは受け継いでいる。

地球環境保全委員会の会議では，Macintoshがつながっているとおぼしきプロジェクタに Directorで作ったとおぼしき画面が出ている。それはともかく，出席者が凄い。黒部進(いうまでもなく，ウルトラマンのハヤタ隊員)，二瓶正也(イデ隊員ね)，桜井浩子(フジ隊員)までいるのだ。いいなあ。いま，ウルトラマンのビデオ見ると，フジ隊員がなかなかいいのだよね。わはは。くすぐってくれる。アンヌ隊員より，当時の面影が偲ばれてうれしい。

が，会議の内容はといえば，単なる地球環境の啓蒙話なのだ。どうしてウルトラセブンでいきなりエコロジー話を聞かされねばならないのか。

案の定，物語は「ウルトラセブン」＋「NHK的啓蒙教育物語」だった。いつからウルトラセブンが政府広報ヒーローになったのだ？ ウルトラセブンはそういったしがらみから自由であり，反骨的シナリオライターを多数抱え，下手をしたら科学文明批判までやってのけるというところが，現在でも視聴に堪えるものになっている要因だったはずだ。

とまあ，ウルトラセブンに思い入れのある人はみな泣いたのであった。なにしろ，番組が終わったら「で，どうでしたか」なんておばさんが出てきて，通産省の人が解説までしちゃうんだから。ウルトラセブンが反体制でなければならない理由はどこにもないけれども，ここまで忠実に「ウルトラセブンして」いながら，政府広報的番組になってしまった悲しさは筆舌に尽くしがたいものがある。かつてのウルトラセブンをそのまま(文字どおりそのまま)現在の技術で作り直した部分と，啓蒙教育的な部分のアンバランスさが致命的であったのだ。

うう。

テレビとパソコンの違いはどこにある

それはさておいて，話はテレビとパソコンの違いへと飛ぶ。テレビ付きパソコンを見ていて思いついたのだ。これらはX1ほどの成功は収めないだろう。なぜなら，X1にはテレビリモコンが付いていたからだ。

すごい結論。

その心はこうである。テレビとパソコンでは，必要とする距離感が異なるのだ。見る人(使う人)とマシンとの距離感だ。パソコンのほうがずっと近い。テレビは仮に14インチサイズであっても1m以上の距離で見るものだ。だが，パソコンはせいぜい40〜50cmで使うものである。パソコンはモニタとユーザーが1対1で向かい合うものであり，常になんらかの操作をユーザー側がするものだからだ。テレビは違う。ユーザーはあくまでも受動専門であり，せいぜい「選択肢をもつ」に過ぎないからだ。ファミコンはその中間的距離かな。

X1であれば，テレビ用リモコンがあったおかげで，どちらの距離感にも即座に対応できた。そうでなくても，キーボードに手を伸ばせばことは足りた。WOODYやPS/V Visionはそうではない。

だから，リモコンをもつX1は「テレビとしても使える」パソコンであり，WOODYやPS/V Visionは「パソコンの合間にテレビを見る」パソコンなのである。ど

ちらのほうが重宝するか。やはり前者であろう(ところが,あとで知ったが,そのへんのところは個人向けパソコンでも実績のある富士通はよくわかっていて，FM TOWNS/fresh TVには専用のリモコンを付けてきた。この点については他機種を1歩リード，というべきか)。

今後出てくるマシンやマルチメディアプレイヤーには,この距離感の違いをうまく克服してもらいたいものである。

メディアになろう

家庭にパソコンが入り，テレビや電話とリンクする。そうすると，何が起きるか。高杉弾はかつて「メディアになりたい」といい，いまは「メディアマン」だといっている。そういうことだ。

将来，1人ひとりがメディアとなるのである。ちょいと話が一足飛びに過ぎるが，まあ，気まぐれな連載ということで許してくださいませ。

すでにパソコン通信では，テキストを通じて，1人ひとりがメディアとなりつつある。全国規模ネットの場合,ROM(Read Only Member：読むだけで発言はしない人)はアクティブユーザーの100倍はいるといわれている。アクティブユーザーが50人いれば，それは5,000人に読まれているということなのだ。凄いでしょ。たとえば,パソコン関係の単行本を書いたとしよう。題材がマイナーなものであれば，初版はだいたい5,000部だ。売れ筋のもので10,000部。よほど売れることが確定している(著名人が書いていたり，売れ線の内容だったり)ものでない限り，初版でそれ以上ということはなく，多くの場合,増刷もない。あくまでも単行本の話ね。

ちなみに，聞くところによると，CD-ROMソフトでもそこそこ売れて5,000〜10,000部。ゲームソフトやビデオソフトでもよく売れて数万のレベルだし，音楽CDだってそうだ。よほど売れ筋でないと，10万以上なんていかないのである。

そう考えてみると，素人のなんてことない書き込みが5,000人以上に読まれるというのは凄いことなのだ。

いまでさえそうなのだから，電話とテレビがつながった時代にどうなるか。たとえば，あなたが何気なく送信した自分の顔が何万人もに見られるのである。全部がつながった時代，当然のようにCDDカメラはパソコンに付いているし，音声なんていまでも当たり前だ。通信ソフトはもっと簡便になり，画像や音声のリアルタイム表示も当たり前。そうなると，あらゆるデジタルデータが簡単に流通できるようになる。CCDカメラによるリアルタイム画像付きチャットなんてのも無理な話ではない。

1人ひとりがメディアになる，という意味がわかってもらえたかしら。

1人ひとりのちょっとした情報発信が何万人もに渡る。重要なのは，情報の有用性ではなく，発信したメディアの個性だ。個性がメディアとなるのだ。

そういう世界では，情報の価値観がガラリと変わる。何しろ，デジタルの世界である。デジタルのなかではみな平等だ。しかも，パソコン通信の世界が「ボツのない投稿欄」といわれているように，そこに流れる情報の質は玉石混淆そのもの。「ボツのない投稿欄」というより「編集者不在の投稿欄」といっていい。雑誌の投稿欄は,編集者が多数ある投稿のなかから取捨選択して載せている。決して，ランダムにピックアップしているわけではない。が，誰も取捨選択できないとなると，どんな情報が流れてくるかまったくわからない。そこから自分にとって価値のある情報を見つけ出すのは，超至難の技だ。いまでもNIFTY-Serveクラスになると，おいしいところだけピックアップして見て回るのは不可能に近い。それだけで1日が終わってしまうほどだ。

1人ひとりがメディアになる時代は，どのメディアが面白いか，どのメディアが役立つかを無限に近い選択肢から選ばねばならない時代なのである。

すでに，書き込みを読みながら，「あ，この人の書き込みならたぶん本当だろう」「この人のは眉に唾付けておこう」「こいつのはウソが多いが面白いから許そう」ってことが起きている。彼らはメディアとなったのだ。

おそらく，自分がメディアであるという自覚をもてるかどうかがひとつのポイントになるだろう。それができていないと，筒井康隆の小説の主人公のように，情報に翻弄され，無数のメディアたちに弄ばれることになる。

ある人が「パソコン通信の世界でも，実社会で起きているありとあらゆる問題が起きると思ったほうがいい」といった。私は「パソコン通信の世界は『ツイン・ピークス』」だと思う。

面白い。

世間では，家庭にマルチメディアが入るときは(もっとも，家庭にマルチメディアが入る，って言葉自体，理解不能だが，まあ，それはおいといて)，パソコンではなく，専用のマルチメディアプレイヤーだろうと思われている。だが，私としてはパソコンの形態で入ってもらいたい。なぜなら，パソコンこそが個人をメディア化するためのツール足りうるからだ。情報を受けるだけではなく，発信できる装置なのである。

はやく，1人ひとりがメディアとなった世界を見てみたいものである。ろくなものではないだろうが，面白そうなことだけは確かだ。とまあ，そういうことである。

このネタは奥が深いので，またいつか，細かくやることになろう。

DOS/Vがくる日まで

Takazawa Kyoko
高沢 恭子

今春から東京と三重県という2つの地に生活の場を広げたキョウコさんですが，さっそく，新しいものや古いものを見たり聞いたりしているようです。どちらも知ると，好奇心はますます広がります。

パソコンの初心者のための本は，いつも私の愛読書になる。ビギナーの本といっても，パソコンの世界はビギナーが身につけるべきことが少しずつ移り変わっていくので，見るたびに新しい。それだけ技術や機械，操作方法が進歩している証拠だけれど，いちばん新しいことをきちんと説明してくれるものがないと，私はパソコン原人のままでとどまってしまうだろう。だから初心者のための本はありがたい。

おもにビギナーのためにつくられて，最近月刊誌になった「Paso」も読んでみた。薄手で軽快なつくりは手にしやすく，どのページも明るくて，誰にでも読めるようにつくったという姿勢がよくわかる。基礎知識のQ＆AやDOS/Vの紹介を読みながら，ずいぶん便利になったものだと一人前なことを思う一方で，パソコン通信をはじめたばかりの人の話題や，マシンのボディを自分のオリジナルデザインで塗装してしまったらどうなるかといった記事などには，10年前の仲間たちがそっくり同じだったことを思い出して感慨深い気持ちにもなった。どの道でも，1人ひとりが自分でたどるところに意味があるのだろう。

アニキの新車

三重のマンションを基地にした週末ごとのドライブで，夫のクルマの走行距離は1万キロほどになった。

クルマが生活の一部に入ってきたことで，私のほうはいままで興味のなかったたくさんのクルマに目がいくようになった。かたはしから車種や製造元，名前をたしかめたり，なんとなく性能まで知ろうとしたりするのだからおもしろい。パソコンをはじめたばかりの人と同じように，クルマがめずらしくてしかたがないのだ。

クルマに乗るともっとおもしろいことがたくさんある。クルマは車体に値段が書いてあるようなものだから，こんなわかりやすい比べっこはない。あまり気にかけていないつもりなのに，大きなクルマ，高級なクルマにはしぜんに目がいく。当然のように，大きなクルマがいばることも多い。その点，人間の価値はクルマほど外見ではわからないのがいいのかなと思ったりする。

自分のクルマを選ぶときは，みなそれぞれの用途や目的，あるいは主義にもとづいて吟味して決めるのだから，見栄や外観を重視することはオロカだと誰だって知っている。それなのに，高価なクルマを持って人に見せたいという気持ちは，これも誰にでもありそうだ。

「とうとう3ナンバーにしたよ」

狛江のアニキが，いま水割りを飲んでいるのだといいながら，うれしそうに電話をかけてきたのもつい先日のことだ。

「日産のラルゴっていうんだけどね。四輪駆動だよ。アクアブルーとイングリッシュグレーと称する色のツートンカラーなんだけど，おマエ，イメージわかるかい？」

相当にうれしくてゴキゲンのようだ。

妻と3人の息子，それに別棟にいる妻の母親と暮らす兄は，一家で遠出することも多く，以前からわりあい多人数乗りの大型のクルマに乗っていた。

「そんなに前とはちがうの？　どんなカタチなの？」と私は聞いた。

「ワンボックスタイプかなぁ，ともかくあこがれの3ナンバーになったよ。見にこないか？」

LARGO(ラルゴ)ってなんだっけ。そうだ，音楽用語だった。辞書にはイタリア語で「きわめてゆるやかに」「ゆったり堂々としたテンポの楽曲」のことだとあった。「遅く，かつ広く」と書いた辞書もある。速さを競うのではなく，威風堂々，ゆったりいこうよなんてなかなかいい。

ところで妹に電話してくるほどハシャぎたくなるところをみると，3ナンバーとやらはそんなにスゴイのだろうか。クルマがスゴイのか，クルマを買った自分がスゴイのか，アニキはたぶん区別がついていないことだろう。

パソコンを買うときも，クルマを買うときと同じだろうか。パソコンをすでに使っている人が何台目かのマシンを選ぶのと，はじめたばかりの人が1台目のマシンを買うのとでは，ちがうだろうか。

この半年くらいのあいだに，夫の会社の人たちが購入したパソコンは，だいたい中古の品が多かった。

かしこいビギナーの輪

三重県で勤務するようになってから，夫が終業後の社内で希望者をあつめて開講していた「パソコン幼稚園」は，昨年の暮れごろ修了した。

講座の名前に安心感があったのか，いつもにぎやかで出席率もよく，成果もそれなりにあったらしい。おそらく，なんとなくチャンスをのがしていたと思われる人たちが何人も，この機会にパソコンを使いはじめたようだ。

教材になっていた「エディタの使いかた」では，いかにすこしのコマンドしか知らなくてもしごとができるかということで，何回も私が登場したそうだ。「家内は10年くらいこれで文章を書いていますが，このロマンドは知りません」。みんな，そのたびにニコニコしたらしい。

愛用の「MIFES」には電話帳ほどのマニ

ュアルがあるらしいが，基本的なコマンドだけで私はすごしてきた。無用なものは身につかないだろうし，必要になったら徹底的に研究すればいい。

「パソコン幼稚園」が意外に愉快にすすめられていったのは，どうも私の考えるところ，教材に使っていた大部分のマシンが，かきあつめた中古のパソコンや夫の持ち合わせのマシンだったという気楽さにあったのではないかと思う。新品の同型のマシンを整列させたカルチャー教室のパソコンレッスンとはだいぶちがう。「園児」のみなさんは，1台ごとにカタチのちがうパソコンでココロおきなくキーを叩いて操作をおぼえ，ついでにいろいろな機種の名前も知ることができた。

その成果か影響かわからないが，パソコンは少しばかり古い型の機種でも性能上なんの不足もないことをみんなが認めた。こんなに申し分ないものが，元の価格の10分の1くらいで買える。もし当たりハズレがあっても，クルマのように生命の危険はない，と思ったかどうか知らないが，はじめて買うパソコンは中古にしようときめた人が多かった。

けっきょく夫はみんなから依頼されたかたちになって，専門のショップや知人の業者H氏から10台以上の中古マシンを調達した。まず1台使ってみて，自分とパソコンの相性をためしてみよう。「幼稚園」の不ぞろいで，ある意味では豊富な教材が，いろいろとヒントをあたえてくれたのだと私は思う。

パソコン歴の長い人，経験をつんだ人が新しくマシンを買うときも，知識がゆたかなだけに迷いもあるだろう。パソコンはクルマのように道路を走ってみんなに見せるチャンスはないが，その代わり税金や車検の義務もないし，使わなくなったマシンの廃車届けもいらない。好きなだけ買い込むことができる。1台買うとすぐ新機種があらわれ，マズイことに経験が長いほど小さな進歩がよくわかる。

パソコンもクルマもその美しさの魅力が大きい。性能が増すと美しさに迫力が加わって見える。それは，内部のメカニズムへの崇拝があるからで，ハコだけの光ではクルマもパソコンも輝いて見えることはない。だからほしいものほど美しいのだろう。

武蔵が走ってる

ドライブで関ヶ原に行った。あの「天下分け目の戦い」の関ヶ原に現実に立つことがあるとは思わなかった。

三重県は奈良も京都も近く，歴史にゆかりの地をおとずれるには，まったく好都合である。日常生活をしながら，いままでイメージの中にしかなかった場所にじっさいに行くことができるのだから，これは予想外の特典といえる。

どこを走っても山と緑にめぐまれた土地柄である。あの桜の季節から新緑の初夏へと，週末ごとに山なみが刻々と青さを増していくのをながめるだけでも，なんとぜいたくなことかと思ってしまう。こういう自然の脈動の大きさに，ほとんど触れることができずにすごしていた東京暮らしの自分は，やはり貧しかった。

関ヶ原はウォーランドと名づけた遊園地ふうの公園になっていた。広い敷地内をかつての東軍，西軍の布陣そのままに縮小して，戦いの装束をした人形を配置し当時を再現していたが，軍旗がはためき，武具も馬もそろい，しとめた生首もならび，これはなかなか迫真ものだった。

帰宅してから夫が吉川英治の「宮本武蔵」をパラパラとめくった。いまの住まいにはほとんどパソコンの書籍しかないのだが，彼は先日帰京したとき，トオルの本棚から文庫本の「武蔵」を何冊か持ってきていた。どうも武蔵の行動軌跡はこのあたりの地域に関わりがあったと記憶していたのだ。

「ああ，やっぱり17歳で又八とふたりで関ヶ原の戦いに出てるよ」

物語の冒頭が関ヶ原戦への参加なのだそうだ。吉川英治といえば，中学生時代の兄が，巌流島でのラストシーンの名文をくりかえし読み聞かせてくれたので，私にはそこだけが鮮明だ。

「雑魚は歌い，雑魚は躍る。けれど，誰か知ろう，百尺下の水の心を」

最後のクダリを感情をこめて読む兄は，陶酔していたものだ。

「ああ，おもしろいなぁ，われわれがおとずれた所に武蔵はつぎつぎ行ってるよ」

乱暴者の武蔵が沢庵和尚にいましめられたり，池田藩の庇護を受けたりしながら剣の修行をつんでいく。ならず者の浪人を切った般若野，吉岡清十郎と出会った柳生の里，追ってきたお通からのがれた伊賀。そのあと伊勢や四日市，鈴鹿峠へと渡っていく。私たちは宮本武蔵が物語の中でかけまわった世界に，いま限られた日々をすごしているらしい。

「ところで，DOS/Vはきまった？」

「そう，DOS/Vね」

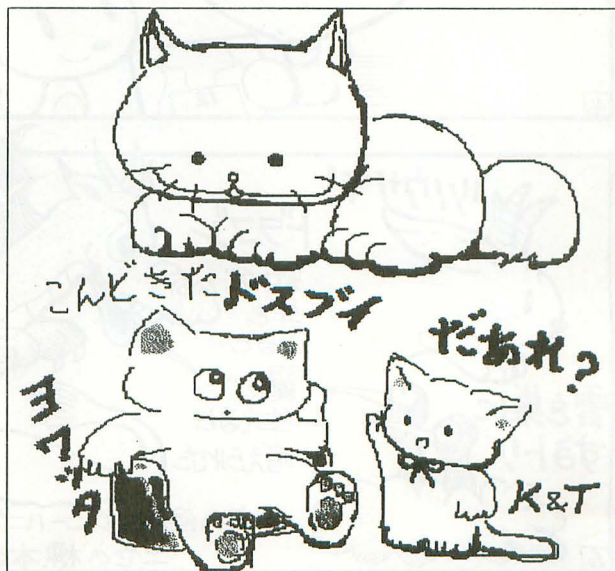
1年くらい前からDOS/Vにすることをずっと検討中の夫なのだ。どんな組み合わせでDOS/Vを取り入れようかと考えているらしいのだが，いそがしいせいかなかなか実現までいかない。

「こんどはイラストもDOS/Vでトライできるからね」といって久しい。

「あの，それよりネ，このあいだ会社の誰かが買ったエプソンのPC-286Lね，あれ，買ってよ，2万円だったでしょ？」

6年前のラップトップマシンだが，30万以上のものだったのだから，2万円なんてタダに近い。当時，国産初の互換機として話題になったそうで，PC-9801UVシリーズとの互換性がある。

「DOS/Vがくるまで，それで遊んで待ってるから，ヨロシクね」

じつは私，ついこのあいだまでDOS/Vってのは，新機種の名前だと思っていた。

illustration：Kyoko Takazawa

another cg world
©KYOKO EGUCHI
GENDAI NO MONSTER

12
怪物誌
MONSTER MAKERS
荒俣宏 編著

12
怪物誌
荒俣宏

12

ごっつう おもろいでっせー
ねェねェ〜
What's?
12

19世紀はじめまではねー
こーんな生き物が
ホントにいるって
思われてたんだってー
当時
写真が
なかったから
ぜんぶイラスト

ツノウサギ
ドラゴン
蛇が
年を
とると
翼が
生えると
考えられた
雲を巣に
するトリ
海の怪物 ジェニーハニヴァー
ニセの標本まであった…

こんな動物や
あんな生き物や もー
いろいろ
あって…
じ〜

な…なにを そんなに
見つめているの？

2頭身
の体

1度も
はずしたことのない
黒メガネ

なんじゃー
それがどーしたってェ
歯に
衣きせぬ
物言い
…

ミトンの手
しかも
ときどきドラエモンの手
にもなる…

しゃべる ウルトラマン
Tシャツ
しゅ
わっち
おまけに 全然 着がえた
ことがない……

いまでも X68000で
CGを作っている
….
ホイ
ホイ
UNIX
マシンより
気楽で
いいや〜

JUNE 26 '94
ねぇ〜
何描いてん
ねん？
これを 現代の
怪獣として記す
べきか否か…
永田町には、
いっぱい
いるけどね…

PENGUIN INFORMATION CORNER
ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

光磁気ディスク
LMO-200/LMO-400
ロジテック

LMO-400

ロジテックは3.5インチ光磁気ディスク「LMO-200」「LMO-400」を発売した。

「LMO-200」は，ディスクの記憶容量が128Mバイト。ディスク回転数は1800rpm。平均シーク速度は120ms。キャッシュメモリを128Kバイト内蔵している。

「LMO-400」は，ディスクの記憶容量が128/230Mバイトのコンパチで，ディスク回転数は3600rpm。平均シーク速度は30ms。キャッシュメモリを256Kバイト内蔵。接続にはどちらも別売りのケーブルが必要となる。

価格は「LMO-200」が79,800円，「LMO-400」が158,000円（それぞれ税別）。

〈問い合わせ先〉

ロジテック㈱　☎0265(74)1455

ハードディスク
LHD-NB340U/LHD-NB520U
ロジテック

ロジテックは2.5インチSCSI-2対応ハードディスク「LHD-NB340U」「LHD-NB520U」を発売した。

「LHD-NB340U」は記憶容量が324Mバイト，平均シーク速度は13ms。キャッシュメモリを128Kバイト内蔵している。

「LHD-NB520U」は記憶容量が498Mバイト，平均シーク速度は12ms。キャッシュメモリを512Kバイトを内蔵している。接続に

LHD-NBシリーズ

はどちらも別売りのケーブルが必要。

価格は「LHD-NB340U」が88,000円，「LHD-NB520U」が108,000円（それぞれ税別）。

〈問い合わせ先〉

ロジテック㈱　☎0265(74)1455

ハードディスク
Areasシリーズ
日本アルトス

Areasシリーズ

日本アルトスはSCSIハードディスク「Areas360DX/AC」「Areas540DX/AC」「Areas1000DX/AC」「Areas1600DX/AC」「Areas2100DX/AC」を発売した。

各機種の記憶容量は上記の順で，360Mバイト，540Mバイト，1Gバイト，1.6Gバイト，2.1Gバイトとなっている。

各機種の平均シーク速度は上記の順で，9.5ms，9.5ms，9.5ms，10ms，10ms。

各機種はキャッシュメモリを内蔵しており，その容量は上記の順で，124Kバイト，512Kバイト，256Kバイト，512Kバイト，1Mバイト。

価格は，「Areas360DX/AC」が89,000円，「Areas540DX/AC」が118,000円，「Areas1000DX/AC」が198,000円，「Areas1600DX/AC」が268,000円，「Areas2100DX/AC」が348,000円（それぞれ税別）。

〈問い合わせ先〉

日本アルトス㈱　☎03(5820)3800

レーザープリンタ
LP-1000/8000SE/9000/9000PS2 F2/F5
セイコーエプソン

LP-1000

セイコーエプソンはレーザープリンタ5機種「LP-1000」「LP-8000SE」「LP-9000」「LP-9000PS2 F2」「LP-9000PS2 F5」を発売する。

「LP-1000」は解像度強化機能（RIT）により，走査線方向の解像度を1200dpi，紙送り方向を300dpiの600dpi相当の画質で印刷が可能になった。アウトラインフォントは明朝体，ゴシック体，毛筆体の3書体を標準装備。用紙サイズはハガキサイズ～A4まで利用できる。また，トナーセーブ機能を使うことで，通常に比べて約50%のトナーを節約することが可能。

「LP-8000SE」の解像度は「LP-1000」と同じ。用紙サイズはA3まで利用できる。インタフェイスにはパラレルとシリアルが1つずつと，拡張スロットが1つ。アウトラインフォントは4書体を標準装備。

「LP-9000」は走査線方向の解像度を2400dpi，紙送り方向600dpiの1200dpi相当の画質を実現。印刷速度はファインモード（1200dpi相当）とクイックモード（600dpi相当）が選択できる。

「LP-9000PS2 F2/F5」の解像度は「LP-

9000」と同じで，印刷速度も選択できる。アドビシステムズ社純正のPostScript Level2ソフトウェアを搭載。アウトラインフォントは日本語2書体（または5書体）と欧文35書体を内蔵している。

価格は「LP-1000」が99,800円，「LP-8000SE」が248,000円，「LP-9000」が348,000円，「LP-9000PS2 F2」が498,000円，「LP-9000PS2 F5」が648,000円（それぞれ税別）。

〈問い合わせ先〉
エプソンインフォメーションセンター
☎0424(99)7133，06(399)1115

マッハジェットプリンタ
MJ-400/MJ-1010
セイコーエプソン

MJ-400

セイコーエプソンはマッハジェットプリンタ「MJ-400」「MJ-1010」を発売した。

「MJ-400」は印字解像度360dpiで，印字速度は漢字全角100cpsを実現している。用紙サイズはハガキサイズとB5～A4に対応。フロントローディングに対応したオートシートフィーダを内蔵し，A4用紙なら100枚の連続印刷が可能。書体は明朝体，ゴシック体，毛筆体の3書体を標準装備。

「MJ-1010」は「MJ-400」の機能に加え，用紙サイズがA3まで可能。印字速度も漢字全角で167cpsを実現している。そのほかに，インタフェイス自動切り替え機能により，最大2種類のパソコンでのプリンタの共有環境を実現できる。ただし，標準装備の書体は明朝体とゴシック体の2種類。

価格は「MJ-400」が49,800円，「MJ-1010」が79,800円（どちらも税別）。

〈問い合わせ先〉
エプソンインフォメーションセンター
☎0424(99)7133，06(399)1115

液晶プロジェクタ
XV-E2Z
シャープ

シャープは液晶プロジェクタ「XV-E2Z」を発売した。

XV-E2Z

同機では，309,120画素の液晶パネルを3枚使用し，水平解像度は500本の画質を再現している。輝度は250Wメタルハライドランプを採用することで，700ルクスを実現した。投写する映像の位置はレンズシフト機能により上下方向約100cmの範囲で調節できる。接続端子は，入力がS端子1系統，ビデオ2系統，ワイヤードリモコン1系統で，出力がモニタ1系統，DC12V1系統となっている。

価格は850,000円（税別）。

〈問い合わせ先〉
シャープ㈱ ☎06(621)1221，03(5261)7271

ビデオ内蔵液晶ビジョン
XV-VM1Z
シャープ

XV-VM1Z

シャープはビデオ内蔵液晶ビジョン「XV-VM1Z」を発売した。

同機は，HiFiビデオ，テレビチューナ，ステレオアンプ，スピーカーを内蔵しており，しかも映像を拡大投写することも，本体モニタでじかに見ることもできる。液晶パネルには301,158画素の3.6型液晶パネルを1枚採用し，水平解像度350本の画質を再現した。液晶プロジェクタとしては，投写距離の設定と合わせ，最小20型から最大120型までの投影が可能。

価格は380,000円（税別）。

〈問い合わせ先〉
シャープ㈱ ☎06(621)1221，03(5261)7271

ビデオプリンタ
VG-100
カシオ計算機

VG-100

カシオ計算機は，家庭用ビデオプリンタ「VG-100」を発売する。

本機はビデオやゲームなどのテレビ画面に映った画像のカラー印刷を行う。画素数は499×682ドットで210dpi相当。色数は209万色，階調制御数は128階調を再現した。1枚当たりの印刷時間は45秒で，印刷の種類も拡大縮小を含む13種類が用意されている。また，テレビの映像を見ながらプリント画面を指定できる子画面表示が可能。

付属品には，ACアダプタ，映像接続ケーブル，ワイヤレスリモコン，用紙50枚，インクリボン1本が用意され，購入してすぐに印刷ができる。

価格は55,000円（税別）。

〈問い合わせ先〉
カシオ計算機㈱ ☎03(3347)4811

INFORMATION

デジタルアート展
VARIETY
ディジタル・イメージ

ディジタル・イメージは5月に東京で行った「VARIETY」の拡大版を開催する。

同展はCGやインタラクティブムービー，CD-ROMなど約170点の作品を展示する。

会場は大阪府立現代美術センターで，期間が8月2日～12日（7日休館），開催時間がAM10：00～PM6：00（6日PM4：00まで）となっている。入場は無料。

また，8月6日，PM1：00～PM4：00には，会場隣の大阪府立文化情報センター多目的ホールにおいて，「草原」と「デジタルイメージ作家」をテーマにシンポジウムが行われる。参加希望者は，現代美術センター（7月26日より受付開始）に電話で申し込むこと。定員あり。

〈問い合わせ先〉
ディジタル・イメージ ☎03(3237)9731
大阪府立現代美術センター ☎06(445)6665

FILES

Oh!X

このインデックスは，タイトル，注記――著者名，誌名，月号，ページで構成されています。暑いからといって，家のなかでゴロゴロせずに外へ遊びにいきましょう。ただ，寝るときには風邪なんかひかないように注意してくださいね。

参考文献
I/O　工学社
ASAHIパソコン　朝日新聞社
ASCII　アスキー
コンプティーク　角川書店
C MAGAZINE　ソフトバンク
電撃王　主婦の友社
PIXEL　図形処理情報センター
マイコンBASIC Magazine　電波新聞社
My Computer Magazine　電波新聞社
LOGIN　アスキー

一般

▶NEWS
ビジネスショウで登場した新製品を紹介する。「DynaBook433」や「PowerBook540c」など。――編集部，ASAHIパソコン，6・15号，8-15pp.

▶特集　パワーをつけろ　今ならこれを買え　夏のパソコン最新主力機種はこれだ
1994年夏のお買い得な機種を選定する。ディスプレイやプリンタなどの周辺機器のガイドもあり。――編集部，ASAHIパソコン，6・15号，16-40pp.

▶機械用言博物館　11
今回は「落ちる」という用語を取り上げ，パソコン内部の事情を解説する。――荻窪圭，ASAHIパソコン，6・15号，104-105pp.

▶THE NEWS FILE
マルチメディアノートパソコン「PC-9821np」，エプソンの「PC-486HA2」など新機種の情報と，「マルチメディア'94 MIDI FAIR TOKYO」のレポートなど。――編集部，LOGIN，12号，112-121pp.

▶架想楽園へ行こうVer.2.02
日本のバーチャルリアリティ研究の現場を訪ねる。今回は富士通研究所。人工生命の研究の現状をレポートする。――中田宏之，LOGIN，12号，162-165pp.

▶New Machine
レーザーアクティブ，3DOの最新動向と，次世代ゲーム機「PlayStation」「SEGASATURN」のニュースなど。――編集部，コンプティーク，7月号，97-99pp.

▶次世代機戦争を完全分析
「PlayStation」「SEGASATURN」「FX」などの最新情報とメーカーへのインタビューで次世代機の展望を明らかにする。――編集部，電撃王，7月号，28-42pp.

▶ゲームクリエイター養成スクールの実態
ゲームスクールの授業内容や就職事情をレポートし，実際に授業を体験。各スクールのガイドも同時に掲載。――編集部，電撃王，7月号，133-150pp.

▶特集　ダイレクト直接購入の手引き
最近サービス内容が充実してきているパソコン通信販売の実態を紹介し，お薦め機種をチェック。――編集部，マイコンBASIC Magazine，7月号，32-44pp.

▶ビジネスショウ'94 TOKYO
東京国際見本市会場で開催された「ビジネスショウ'94」の模様をレポートする。――編集部，マイコンBASIC Magazine，7月号，50-51pp.

▶CD-ROMからはじめるマルチメディア
傷がついてしまったCD-ROMを補修するキットを紹介する。――吉岡哲也，マイコンBASIC Magazine，7月号，54-55pp.

▶特集　見えてきた新ハード
次世代ゲーム機の情報をまとめ，各社の動きを報告する。――山下章，マイコンBASIC Magazine，7月号，139-145pp.

▶Arcade Game Graffiti　第5回
今回は1980年のアーケードゲーム市場の模様だ。「クレイジークライマー」などが登場。――編集部，マイコンBASIC Magazine，7月号，152-155pp.

▶NEWS
各社の次世代ゲーム機やIBMから発売されるA5版カラーブックのニュースなど。――編集部，ASAHIパソコン，7・1号，8-9pp.

▶光磁気ディスクはハードディスクを超えるか
MOセットアップ術や，そのほかのメディア（リムーバブルHDなど）との比較をもとにMOの将来性を検証。――編集部，ASAHIパソコン，7・1号，120-129pp.

▶機械用言博物館　12
「クラッシュする」という用語を取り上げ，システムの保守などに言及する。――荻窪圭，ASAHIパソコン，7・1号，132-133pp.

▶ビジネスショウ'94 TOKYO
5月17日から20日まで行われたビジネスショウをレポートする。――編集部，I/O，7月号，32-33pp.

▶特集2　周辺機器先取り情報
ハードディスク，CD-ROMドライブ，MOドライブ，モデムに焦点を当て，各社の製品を紹介する。――編集部，I/O，7月号，97-106pp.

▶MOTOROLA 68060
モトローラ社が開発した68000ファミリーの最新プロセッサ「MC68060」の特徴を紹介。――編集部，I/O，7月号，112p.

▶PowerPC 604
IBMとモトローラが発表した「PowerPC 604」の特徴を紹介する。――編集部，I/O，7月号，113p.

▶小特集　MIDI・GM音源購入術
音源とMIDI音源ボード，ソフトまで含めたDTM志向のGM音源購入術を紹介――高橋克行，I/O，7月号，114-122pp.

▶MultiMedia Watching 7
今回はISDNカラオケがマルチメディアかどうかについて考える。――編集部，I/O，7月号，127-130pp.

▶ASCII EXPRESS
「COMDEX Spring'94」「ビジネスショウ'94 TOKYO」などのショウレポートや新型ザウルスなどの新製品を紹介する。――編集部，ASCII，7月号，246-268pp.

▶夏の最新機種情報
Pentium90を搭載しPCIバスを採用したマシンから，テレビ内蔵マシンまで最新機種を徹底紹介。――編集部，ASCII，7月号，269-300pp.

▶特集　対決！　ATOK8 vs VJE-Delta
業界を代表する2大日本語入力システムをさまざまな観点から徹底比較していく。――編集部・箭内敏夫・金城孝吉，ASCII，7月号，309-324pp.

▶INTERCOOLED
ソニーの「PlayStation」，セガの「SEGASATURN」「SUPER32X」に関する情報や3DOのゲームソフトの紹介など。――編集部，ASCII，7月号，370-375pp.

▶魅惑のニューテクロノジー
EDRAM, SDRAM, CDRAM, RDRAMを取り上げ，高速DRAMの技術について考える。――編集部，ASCII，7月号，380-385pp.

▶稀代もののけ考
「バカパパのモノを買い物」のタイトルが変わった。今回は初夏に楽しむアウトドア遊びグッズを紹介。――バカパパ，ASCII，7月号，464-465pp.

▶特集1　最新FAXモデムの活用と選び方
FAXモデムの原理と導入の仕方を解説。また，機種ごとの特徴も紹介する。――編集部，My Computer Magazine，7月号，15-29pp.

▶特集2 MO「光磁気ディスク」導入のすすめ
MOディスクの構造や仕組みの解説や，セットアップ術などを紹介する。――編集部，My Computer Magazine,7月号，30-49pp.

▶PC New Product
エプソンのプリンタ「MJ-700V2C」やシャープの新型ザウルスなど最新ハードを紹介する。――編集部，My Computer Magazine，7月号，75-83pp.

▶パソコン研究室
今回はフロッピーディスクの製造工程とミクロの世界について探る。――Space Club，My Computer Magazine，7月号，121-123pp

▶ビジネスマンのための情報管理術
「ザウルスアイリス」を利用してザウルスとデータベースを連動させる。――塚田洋一，My Computer Magazine，7月号，174-177pp.

▶特集　パソコンがっちり買いましョー
時代の流れに乗ったパソコン購入術を教える。本体だけでなくアクセラレータボードなどの周辺機器もあわせて紹介。――編集部，LOGIN，13号，123-139pp.

▶THE NEWS FILE
NEO・GEO CDの発表やビデオCDソフト販売開始のニュースなど。最新の情報をピックアップ。――編集部，LOGIN，13号，140-147pp.

▶次世代ゲーム機は世界をゆるがすか!?
「SEGA SATURN」と「Play Station」のスペックを徹底比較。開発中のゲームもリストアップ。――編集部，LOGIN，13号，176-181pp.

▶ECTS探訪記
ロンドンで行われたECTSの最新レポート。主にパソコ

ンのニューソフトをピックアップ――編集部, LOGIN,13号, 182-185pp.

▶くねくね科学探検　第１回

連載第１回は東京大学工学部の原島博教授の研究, "顔"についてレポートする。――鹿野司, LOGIN, 13号, 192-195pp.

▶特集　バーチャルリアリティの実用化を模索する

各所で研究されているVRの映像部分に焦点を当て, VRの実用化を考える。IVR'94に関する情報つき。――編集部ほか, LOGIN, 13号, 70-83pp.

X1/turbo/Z

X1シリーズ

▶FREEZE! PIV

２つの天使像を合わせて消していくアクションパズルゲーム。――charlie, マイコンBASIC Magazine, 7月号, 114-115pp.

X68000

▶Release Data

新作ゲームソフトの発売情報。X68000用は「宝魔ハンターライム」など。――編集部, LOGIN, 12号, 8-9pp.

▶NEWSOFT

「麻雀航海記」ほか, 各機種用の新作ソフトを紹介する。――編集部, LOGIN, 12号, 10-25pp.

▶X68新聞

SX-WINDOWのバージョンアップ記念にGUIについて考える。――編集部, LOGIN, 12号, 134-135pp.

▶未確認クリエイターズ

ユーザーの投稿ゲーム「ガーディアンズ」を紹介する。――編集部, LOGIN, 12号, 142-147pp.

▶'93コンプティークSOFT大賞

1993年に発売されたソフトからベスト１を読者投票で選ぶ。X68000版「ストリートファイターII'」がアクション部門賞を受賞,「ぷよぷよ」もETC部門賞に輝く。――編集部, コンプティーク, 7月号, 20-26pp.

▶SUPER SOFT EXPRESS

シャープの新製品情報ディスク「EXEディスク」や「あすか120%　Burning Fest.」などが登場。――編集部, コンプティーク, 7月号, 27-47pp.

▶新作王

新作紹介コーナー。X68000用は「Mr.Do!/Mr.Do! VS UNICORNS」が登場。――編集部, 電撃王, 7月号, 183-209pp.

▶ホッピングdeあきかんつぶし

ホッピングに乗っている少年を操作して, 地面上を転がっている空き缶をつぶすゲーム。――高橋潤, マイコンBASIC Magazine, 7月号, 116-117pp.

▶花よりダンゴ

天秤を使い荷物を移動するパズルゲーム。――藤井勝敏, マイコンBASIC Magazine, 7月号, 118-120pp.

▶風の伝説ザナドゥ～風の伝説～

ミュージックプログラム。アレンジあり。――重長孝之, マイコンBASIC Magazine, 7月号, 125-129pp.

▶SUPER SOFT HOT INFORMATION

X68000には「雀神クエスト」「Mr.Do!/Mr.Do! VS UNICORNS」が登場。――編集部, マイコンBASIC Magazine, 7月号, とじこみ付録２-38pp.

▶AV STRASSE

「SX-WINDOWver.3.1」, ビデオ入力ユニット「CZ-6VS1」,「MUSIC SX-68K」を紹介する。――中村康秀, ASCII, 7月号, 397-400pp.

▶ONLINE SOFTWARE INDEX

大手ネットにアップロードされたソフトを紹介する。X68000用はベクトルフォントの編集が可能なエディタ「Zeitor.x」など。――編集部, ASCII, 7月号, 509p.

▶なんでもＱ＆Ａ

「シャーペン. X」の用紙設定の方法や, ビデオ入力ユニット「CZ-6VS1」などに関する疑問に答える。――編集部, My Computer Magazine, 7月号, 190-191pp.

▶HOBBY EXPRESS

X68000用では「大魔界村」「The World of X68000 II」を紹介する。――編集部, My Computer Magazine, 7月号, 203,207pp.

▶NEW SOFT

ビデオゲームアンソロジーシリーズVol.10の「Mr.Do!/Mr.Do! VS UNICORNS」を紹介。――編集部, LOGIN, 13号, 21pp.

▶GAME BUSTERS！

「スーパーリアル麻雀PIV」を紹介。――編集部, LOGIN, 13号, 244-245pp.

▶Game Review

少し前に発売されたゲームを再レビューする。「マッドストーカー」が登場。――編集部, LOGIN, 13号, 248-250pp.

▶SX-WINDOWプログラミング　第９回

今回はウィンドウのリサイズとSX-WINDOW開発キットのガイドに従ってプログラムの変更を行う。――吉野智興, C MAGAZINE, 7月号, 124-131pp.

ポケコン

PC-E500

▶15PAZZLE

ポケコン上で遊べる15パズル。――谷口淳史, マイコンBASIC Magazine, 7月号, 121p.

新刊書案内

ヴァーチャリアン嘘つかない
～マルチメディアの正体を暴く～
渡辺浩弐著
主婦の友社刊
☎03(3294)1137
A5変型判　173ページ
1,000円（税込）

渡辺浩弐がゲーム誌で連載していた内容をまとめたもの。渡辺浩弐は「GTV」の編集長。最近では「SPA!」のハイテクヒッピーズにも連載しているゲーム畑の人だ。そのゲーム業界の人がマルチメディアについて書いた。大人たち(マルチメディアという言葉に便乗して商売しようとしている人たち)は何もわかっちゃいない,ってことを延々と書いたのだ。ゲーム誌での連載であるから, 平易でノリのいい言葉で書いている。その視点は非常に正しい。マルチメディアとは「モニタでなんでもできちゃうこと」といい切ってしまうのだから。しかも, いい加減なことを書いているように見えて, そのくせ, しっかりと, 任天堂の宮本茂氏とか, セガの鈴木裕氏とか, 大学のセンセイとか, ブルセラマニアだとか, AV女優だとかに会って, 話を煮詰めている。体験談を挿入することで, 抽象的な言葉を使わずに信憑性を上げているわけだ。マルチメディアによって, さまざまな大人たちが「画質や音質を上げること」に血眼になっている。が, そんなことはどうでもいいと叫ぶ。『「脳」的リアリティ』こそがゲームにとって重要だと。「テトリス」が何故面白いか。『「現実世界」をデジタル化して収録」することではなく,『「ゲーム的な別世界」を作り込む』ことだという。これはおそらくゲームに限った話ではない。

本書の次のポイントは「メディアの違いを理解すること」だ。ブルセラショップで売っているパンティはすでに「モノ」ではなく「メディア」なのだということ。小津の映画は映画館で見ると最高だけど, ビデオで見るとつまらないということ。

巻末の「マルチメディア<的>ゲームソフト10選」を含めて,「ゲーム世代」からのマルチメディア論を展開している。先端技術に関する考察はこなれていない感じもあったが, 偉い人が書く小難しいマルチメディア啓蒙書を読むくらいなら, これを一読すべきである。（K）

ステレオ
感覚のメディア史
吉村　信
細馬宏通編著
ペヨトル工房刊
☎03(3847)0987
四六判　254ページ
2,500円（税込）

昨年, ステレオグラムが大流行した。あれが, 見えたときは気持ちいいものである。

本書では映像に関してのステレオに焦点を当てて, ステレオの歴史, 立体視を助けるために発明された道具の解説, ステレオのもつメディア性, 社会への影響などについて触れている。ほかにも, ステレオ写真や視覚心理学についての対談をとおして, ステレオ文化について考察している。おまけに, ステレオ写真の撮影方法つき。

ステレオグラムが, ただ見えた見えないで一喜一憂するのではなく, もう一歩進んでその感覚がなぜ気持ちいいのか考えてみるのもいいだろう。

スイス時計紀行
香山知子著
東京書籍刊
☎03(5390)7531
四六判　224ページ
1,400円（税込）

あなたは, 時刻をどうやって知るだろうか。おなかの空き具合から, それとも日の高さから, でもほとんどの人は時計から時刻を知るだろう。それも自分の腕時計から……。

本書では, 主にスイス時計の歴史や現況が, 時計に関するエピソードを交え, 興味深く書かれている。たとえば, 永久に動き続ける"自動巻腕時計"や水の中でも動く"防水時計"開発, 戦争との関係, そして"スウォッチ"についてなどいろいろである。

いつもは何気なくしている腕時計に興味をもたせてくれる１冊である。

QUESTION and ANSWER

GCCでプログラムを作っているとき，できあがるプログラムのスタックの大きさを指定するにはどうしたらよいのでしょうか。また，プログラムに最適なスタックサイズを求めるにはどうすればよいのでしょうか。

東京都　岩下　克也

初めは小規模だったプログラムがだんだんと成長していくうち，ある時点から突然動作が不安定になったとします。しかも，直前に変更したソースとはまったく関係のない部分で不安定だとします。こういう場合の原因には，スタック不足が考えられます。

XC，GCCでは関数の引数の受け渡しや局所変数などにスタックを使います。

リスト1をコンパイルして，できた実行ファイルをダンプしてみましょう。後ろのほうに_STACK_SIZEという文字列が見えると思います。この文字列の直前4バイトがこの実行ファイルのスタックサイズです(図1)。この場合，64Kバイトになります。しかしリスト1では31Kバイト×5もの局所変数を取ろうとするので，この実行ファイルはうまく動かないでしょう。私の環境で実行してみたところ，一見正常に終了しますが，次になにかするとシェルがアドレスエラーになってしまいました。こういった場合，スタックサイズを適切に指定する必要があります。

XC ver.2.0ではcc -Gs256kなどとすることでスタックサイズを指定することができます。しかしGCCにはそういったオプションが存在しません。

スタックサイズというのは，実はコンパイラが決めているのではなく，ライブラリが決めているのです。プログラマが書いたmain()にくる前のスタートアップルーチンでスタックが確保されます。スタートアップはXC2のライブラリならCLIB.L，libcならばlibc.aに入っています。これらスタートアップのオブジェクトには「スタックの大きさは識別子_STACK_SIZEの値である」という情報が入っています。リンクするとき，どのオブジェクトファイルにも_STACK_SIZEが定義されていなければ，CLIB.Lまたはlibc.aの中に用意されている，_STACK_SIZEの値を定義したオブジェクトをリンクします。

XC ver.2.0の-Gsオプションは，コンパイルするときに指定された値で_STACK_SIZEを定義します。リスト1をcc -Gs512K -Fs list1.cとしてできあがるlist1.sの最後には_STACK_SIZEの定義があります。ここで定義されているので，ライブラリ中にあるスタックの大きさのデフォルトの値は採用されないわけです。

ここからが質問に対する回答になります。

_STACK_SIZEという識別子を希望の値で定義すればいいということはわかったと思います。その方法を3通り挙げます。

1)　オブジェクトを用意する

```
    .global _STACK_SIZE
_STACK_SIZE equ 256*1024
    .end
```

このソースをアセンブルします。実行ファイルを作るときには，ここでできたオブジェクトファイルを一緒にリンクします。

2)　インラインアセンブラを使う

Cのソースファイルの先頭にでも，

```
__asm ("¥t . global _STACK_SIZE¥n"
  "_STACK_SIZE equ 256 * 1024¥n");
```

と書いておきます。

3)　リンカで定義する

GCCを使っているなら，きっとHLK.X(そると氏作)も使っていることでしょう。HLK.XにはLK.Xにはない便利な機能があります。

```
hlk -d_STACK_SIZE=40000 list1.o -l libc.a libgnu.a
```

とすることにより，_STACK_SIZEを定義できます。=の後ろの数値は16進数です。

スタックのほかに，ヒープの大きさも同様にして指定します。識別子が_HEAP SIZEとなります。

ヒープというのはmalloc()で確保する領域のもとになる領域です。XC2のライブラリではヒープの大きさの初期値が64Kバイトなので，これより大きな領域をmalloc()するにはsbrk()でヒープを増やす必要があります。libcでも初期値は64Kバイトですが，足りなくなると自動的にヒープが伸びるのであまり気にする必要はありません。

ちなみに，すでにできあがっている実行ファイルでも，オプションによってスタックとヒープの大きさを指定できます。XC ver.2.0のライブラリを使用している実行ファイルなら，

-stack:数値 -heap:数値

libcを使用している実行ファイルならば，

-+-s:数値 -+-h:数値

をコマンドラインに付加します。これらのオプションはmain(argc，argv)には渡されません。

スタックの大きさをどれぐらいにするのが適当なのかは，なかなか難しい問題です。局所変数と関数の引数をすべて予測しただけではダメで，利用しているIOCSコール，DOSコール，SXコール内部のことまで考えなければなりません。私見と経験では，局所変数のぶん+8Kバイトぐらいあればいいと思います。

XCなら，

cc -Gc

GCCなら，

gcc -fstack-check

とすることにより，実行時にスタックから局所変数が取れるかどうかをチェックするようになります。DOSコールなどの中までチェックしませんが，ある程度の有用性はあるでしょう。

特に，再帰呼び出しをするようなプログラムでは，できるだけ-fstack-checkをつ

図1

```
00001F70  66 6D 74 6F 75 74 00 00 02 04 00 00 1F A0 5F 65 fmtout......._e
00001F80  6E 76 69 72 6F 6E 00 00 02 00 00 01 00 00 5F 53 nviron......._S
00001F90  54 41 43 4B 5F 53 49 5A 45 00 02 04 00 00 1F A4 TACK_SIZE.......

スタックの大きさは 0x00010000 == 64K Byte
```

リスト1

```
/* スタックが足りないプログラム by けんと */
#include <stdio.h>

long foo( long i, char* ac ) {
    char b[31*1024];
    if ( !i ) {
        return 0;
    } else {
        return foo( i-1, b );
    }
}
int main( void ) {
    char a[31*1024];
    foo( 4, a );
    fprintf( stderr, "finished.¥n" );
    return 0;
}
```

けてコンパイルするようにしましょう。
(田村　健人)

これまでずっとWP.Xを使って文書を書いていたのですが，SX-WINDOW ver.3.1のシャーペンが非常に使いやすいので最近そちらに移行しました。そこでちょっと困っているのがスペースの入力です。文章中でスペースを入力したいときにキーを押すと，スペースが入力される代わりにASK68Kがそこまで入力していた文字列の変換動作に入ってしまいます。ですから，いちいちそれらの文字列を確定してやらないとスペースが入力できません。ASK68Kのコンフィグファイルから SP (スペースキー) の指定してある行を削ってみましたが，なぜか症状は変わりません。どうなっているのでしょうか。
岩手県　島崎　通夫

確かにWP.Xの無変換モードではスペースキーは変換キーとして扱われずにそのまま打ち込むことができました。しかし，標準のASK68K ver.3のキー設定ではスペースキーが変換キー扱いされていますのでWP.X以外ではこのようなことはできませんでした。

これは変換キーの割り当てを変えれば解決する問題なのですが，ASK68Kのコンフィグファイルの設定は，該当する項目を削ってもデフォルトの設定が残っているみたいですので，確実に別の割り当てにしないかぎり変えることはできません。

具体的には，

```
ENTER2=NULL
NEXTKOUHO2=NULL
```

のように明示的にカットしてください。

また，キー割り当てを変更しない場合，単にスペースキーを押すと変換にいってしまいますが，コントロールキーなどを併用すれば直接スペースを入力することは可能になっています。

SX-WINDOWを使っています。ウィンドウが煩雑に開くので最近はよく使うファイルのアイコンをデスクトップに置いて作業することが多いのですが，たまにそのファイルのあるディレクトリが開きたくなることがあります。アイコンをつまんで持っていくと，親ディレクトリを開いてくれるツールとかって作ってもらえませんか？
宮崎県　横田　正則

ツールを作るまでもありません。メニューメンテで，そのアイコンに使われているであろうメニューの9番 (または10番) をエディットしてください。

メニューメンテを起動して，メニューの項目一覧のうち，いちばん下にある空白の部分をクリックすると新規メニュー項目が設定できますので，

項目名
　ディレクトリを開く
実行ファイル
　DI.R

のような指定を加えてください。実行オプションは無指定のままでかまいません。これで登録ボタンを押せば横田さんのお望みの機能が実現されます。

ちなみに，ここで指定したDI.Rというのはディレクトリウィンドウを開くためのSX-WINDOWの内部コマンドです。ファイルセレクタとして出回っているのDI.Rとはなんの関係もありません。

確か68040にはFPU (浮動小数点演算ユニット) が内蔵されていますが，これは68882の命令を全部はサポートしておらず，それに対応するための基本ルーチンがモトローラから提供されるということになっていたと思います。本題ですが，040turboにはモトローラの演算ルーチンがドライバとして入っているのでしょうか。それとも68882用にコンパイルされたオブジェクトなどは動かない仕様のものなのでしょうか。モトローラの資料ではソフトで処理しても68882より高速となっていますが，実際にはどうなのでしょうか。
静岡県　山下　恭慈

68040のFPUは簡易型で，加減乗除などの基本的な演算命令は内蔵していますが，関数演算などはサポートされていません。これらはソフトウェアで処理しなければなりません。

ご指摘のとおり，モトローラからそのためのソフトが提供されることになっているはずですが，040turboで使っているものは独自に作成されたもののようです。

040turboにはフリーソフトウェアでPFLOAT40.Xというドライバが用意されており(中村ちゃぷに氏作成)，これを組み込んでおけば68882用にコンパイルされたオブジェクトでもそのまま処理できるようになります。

ちなみに，68040でサポートされている命令は，FMOVE, FADD, FSUB, FMUL, FDIV, FSQRT, FABS, FNEG, FMOVEM, FCMP, FSAVE, FRESTOREの12種類です。これらの命令に限っていえば，加減算や乗算なら倍精度実数で6～10クロック，除算は41.5～43.5クロックの暫定実行速度となっています。68030の整数乗算が44クロックでしたから，スケールの違いがわかるでしょう。

最終的な速度は関数の使用頻度次第ですが，概ねなにをやっても68882より高速だと思ってもよいようです。基本演算が馬鹿っ速いのは確かですので，浮動小数点演算を多用するプログラムは格段に速くなります。
(中野修一)

最近，質問箱への質問がちょっと少なくなってきました。愛読者はがきから質問を探したり，古い質問はがきを掘り返したりと，担当スタッフ一同困っていますので，日頃なにか疑問に思うことなどがありましたら，なんでもかまいませんから遠慮なく送ってきてください。回答者を指名したいときには希望する回答者名を併記してもらってもかまいません。希望どおりにいくとは限りませんが，できる限り対応したいと思います。それではひとつよろしくお願いします。
(U)

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を挙げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので電話番号も明記してください。

宛先：〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社出版部
Oh!X編集部「Oh!X質問箱」係

FROM READERS TO THE EDITOR

真っ青な空を見上げて胸一杯に空気を吸い込む。とっても大きな入道雲がいまにも落ちてきそう。夏の日光を体の正面で受け止め，ここぞとばかりに動きまわる。でも，無理は禁物。たまには涼しい木陰でお昼寝するのもいいかもね。

◆6月号の特集は，一般的で実用的なことだったので，素人の僕でも，理解できて楽しく読めました。あまり特化した内容ではなく，このような幅広い内容もたまにはよいのではと思いました。　黒木　恒(21)神奈川県

◆「メガディスプレイへの道」が最高でした。用意された解像度をそのまま使う「受け身」とは違う，挑戦的(!?)なディスプレイの使い方をしていると思いました。いやはやマルチスキャンCRTを買ったかいがありましたよ。ちなみに，αデータの「ACM-217」というフラットスクリーン，ノングレアのもので，17インチです。お値段はコネクタつきで84,000円でした。X68000の31kHzモードはあまり大きく伸ばせないのですが，少なくともCZ-6＊＊よりはキレイです。なかなかよい買い物をしました。
安井　百合江(19)愛知県

◆Oh!Xに載っていた「CRT960」を自分で24kHzモードに対応して無理矢理使っていましたが，モニタを選ばなければもっといろいろできるんですね。ちょっと感動してしまいました。
天達　雄一(19)広島県

6月号の特集はなかなか好評でした。苦労していた瀧氏も，送られてくるアンケートハガキを読んで喜んでいました。

◆いやー，ついにやりましたよ。連動コンセントを作ることに成功しました。私みたいなド素人にもできるものですね。最初は電源を入れた途端に燃えだしたらいけないと思って，水の入ったバケツを用意してから(本当)，恐る恐る電源を入れましたが，何事もなく私のディスプレイはX68000と連動してくれました。もー感動！瀧様，私に作る喜びを与えてくださってありがとうございます。今後も「あくまでもローテク」で，がんばってください。
加藤　和人(18)愛媛県

完成おめでとうございます。これからも，連載は続きますので，いろいろと挑戦してみてくださいね。

◆ほんのちょっとアクセラレータの記事が出ました。嬉しいです。石上さんは寮住まいのようですが，就職されたのでしょうか。もしそうなら，制作は難しいかもしれませんね。会社というところは個人を潰しにかかりますから……。「フレーフレー石上」「ガンバレガンバレ石上」
横山　紘一(49)埼玉県

今月号にようやく続報を載せることができました。皆さんのたくさんの声援ありがとうございました。それにしても，「会社というところは……」の部分は年齢の重さを感じさせます。

◆府立高専DōGA研究部(仮)設立のため部員募集……とはいい出したものの1人も部員が集まらない。もともとこのクラブはアニ研みたいなことをするようだった。その話を聞いたオレが「こんなんを作るクラブにしないか」と，持っていたOh!X 4月号の「アマチュアCGAコンテスト」の記事を見せた。すると，クラブの活動目的がこっちに変わったのだ。クラブとして活動できるようになってから作品を作ろうと思う。多分，クラブになるのに2年かかると思うけど(ウチの学校のクラブの基準は2学科，3学年にまたがる部員，ちなみに僕らは1年生)。
福間　雅弘(16)大阪府

クラブになってからなんていわず，できることから始めてみてくださいね。

◆嬉しー！　MOがつながった。しかも230Mバイトタイプだ(実際は217Mバイト程度)。キャラベルデータシステムの「PS-230MO/P」だけど，SxSIで登録するだけで，簡単バッチリ。というわけで，ハードディスクの整理をしています。
木村　守(24)東京都

ハードディスクの整理だけでなく，MOのバックアップも忘れずにとりましょうね。なまじフロッピーディスクより容量が大きいだけにデータを失ったときの悲しみはそれはもう……。

◆ついにHDクラッシュを体験してしまいました。復旧に4，5日かかって，もうヘとヘとでした。そこで，ついにMOを購入。これでママも安心さ！　鈴木　寛之(21)千葉県

ここに，ストリーマを買ってくれればパパも安心(？)。

◆いつになったらX68000の新機種が出るんだろう。もしかしたら今年の新機種ってビデオ入力ユニットのことじゃないだろうか(68020入ってるし)。P.S.「地底最大の作戦」に思わずナミダが……。その昔，このゲームを遊びたくて，MZ-700に移植しようとしたけど，うまくいかなかった思い出がー。　高松　英明(25)香川県

涙に溺れていないで次はあなたの番ですよ。SX-BASICでの投稿をお待ちしております。

◆「猫とコンピュータ」がよかったです。僕は高校の3年間，年末年始の郵便配達のアルバイトをしていました。街に住む人々の生活を肌でじかに感じられた楽しい体験でした。皆さんも1回どうです。あっ，郵便局の回し者じゃありません。　川田　宏(19)香川県

会社がバイトが禁止なものですから……。

◆5月18日，我家のX68030も満1歳を迎え，ますます快調に働いています。周辺機器やメモリ，コプロもと考えていますがお金が……。横でMacintoshも……。金喰いマシンが騒いでいる！
尾形　淳一(41)北海道

X68030は健全なる成長を遂げていますでしょうか？　Macintoshとともに可愛がっ

◀岡村　直也　兵庫県
X68000を所有している2人の仲は，大接近？　あと，気にされてる(？)イラスト大賞の集計は前年4～当年3月号なので11枚であってますよね。

◀加藤　信夫　宮城県
バーチャファイターのイラストは初めてですが，髪の感じがグッド。それにしても，最近のプリンタって本当にキレイに印刷できますよね。

てあげてくださいね。

◆先日，授業のレポートを最近手に入れたTeXを使って提出した。すると先生が「ほう。これはTeXですか？」と尋ねられ，「はい」と答えると，「機種はなんですか？」という問いに，「マイナーだから聞かないでください」と答えた。もう胸を張って「X680x0です」といえないことに情けないものを感じた。辻　圭右(20)東京都

あなたの気持ち次第だと思うのですが……。ちょっと寂しい。

◆うちのX68000君は，完全にゲームマシンと化してしまった。さすがに，会社で8時間以上もディスプレイを見続けると，家に帰ってまで見る気はしなくなると思っていたが，それでも土，日はジョイパッドを握って「餓狼伝説2」をしている。ああ，結局，パソコン少年は成長してN社系列の会社へ入って，S社から離れても，やはり，S社系パソコン少年に変わりはなかった。　西村　英章(24)静岡県

そろそろ，少年とはいえなくなる歳のような気が……。

◆テレビでFM TOWNSⅡ Fresh・TVのCMをやっていた。「パソコンでテレビも見れなくっちゃ」だそうだ。シャープを差し置いて，こんなCMをやるとは……許せん！　シャープさん，ぜひとも本家の意地を見せてやってください。

岡田　耕一(19)山口県

IBM，富士通，松下電器と3社でテレビの映るパソコンが出ましたが，テレビだけについていえば松下電器のWOODYがウィンドウ内に小さくしておける点がいいと思うのですが……。ちなみにX1は1983年発売。ふう。

◆店に並ぶOh!Xの数が増えました。半年くらい前はすぐ売り切れになりましたが，増えたため，数日(2日位)遅れても買えるようになりました。ユーザーが増えたのかな。

虫上　知弘(42)神奈川県

◆6月号が近所の本屋に入荷していなかったのでドキッとしました。　鈴木　康夫(26)千葉県

相反する2枚のハガキ。結果的に発行部数は増えたのでしょうか？　それは秘密です(古い)。

◆最近，ほとんどプログラミングをしていない。入力中の苦しみとデバッグの発狂，思いどおりに動いたときの至福の喜び。忙しくてその時間もないし，現在必要なプログラムというのもない。しかし，プログラミングを「まったく」しない生活というのもなにか充実感がないものである。　砂原　弘幸(22)千葉県

それでは，人助けだと思って，Oh!Xから作ってほしいプログラムを連絡しましょう。時間はそちらで作ってくださいね(冗談)。

◆某店の中古ソフトには「リサイクル」シールが貼ってある。これもなんか納得できない。

久米　豊信(25)大阪府

実はパッケージがリサイクルされていて，中身はまったく関係のないソフトが入っていたりして……。

▲吉田　淳一　宮城県
彼女が着ているのは制服でしょうか？　素朴な感じがいいですね。右手はちょっと気になりますが、顔の感じはけっこう好きなのでがんばってね。

▲山下　彰　神奈川県
この女の子は松井のファンでしょうか。ところで、松井といえばインタビューでゴジラといわれるたびにムッとしているような気がしませんか？

◆友人に「パソコン買うならなにかな？」と聞かれ，とりあえず，いま一番いいのは，買わないことだといったのですが，X68000ユーザーにあるまじき行為でしょうか？

岡　誠治(18)大阪府

あなたがそう判断したのであれば，それでもいいのですが，やっぱり100万台が……(しつこい？)。

◆「スーパーリアル麻雀PⅣ」を買った。アーケードでは香織に3回しか勝てなくて最後までいけなかったけれど，X68000では最後までいけた。ところで，香織のあとの○○○はパソコンのオリジナルですか？　それを確かめるためにアーケードをやっても香織に勝てずじまい。編集部の人で知っている人は教えてください。

寺田　朝成(21)愛知県

アーケード版にもあったと思います。というのは，実際にはそこまで行けたことがないので……。だって，キツイ。

◆歯医者に通っています。老人の歯は1本治せば1本だめになるの繰り返しです。総入れ歯になるまで続きそうです。そういえば，X68000ACEの内蔵HDが戦死しました。いつかフタをあけて葬式をしてやろう。いまはマーラーの「亡き子をしのぶ歌」など聞きながら，昇天したデータを懐かしんでおります。それにしても早死にだったが，ニコチン中毒かもしれないな。

藤田　智(69)静岡県

いまは新たな母(外付けハードディスク)がたくさんの子供(データ)を育てているのでしょうか？　今度はタバコの煙には気をつけてあげてくださいね。

◆先日，NS高輪ビルの前を通りました。“たかわ”だと思ってましたが，“たかなわ”なのですね。　飯嶋　徹(22)神奈川県

地名には独特な読み方をするものが多いですよね。間違った読み方を覚えてしまい，人との会話のなかで恥をかいてしまいます。えっ，私だけですか？

◆予備校に通いはじめて1カ月がたった。なんか，最近の若い人にありがちな，礼儀のなさにはあきれてしまう。ぶつかっておいて謝らずにいってしまうし，通路をふさいで立ち話をして，後ろに人がきてもどこうともしない。勉強だけできれば，あとはなにをしてもいいと思ってるのかなあ？　高校を卒業してから，働いていたので，3歳ぐらい年のはなれた人たちと勉強してるはずなのに，なんだか負けてばかりで，情けない日々。　内田　淳(21)静岡県

勉強ってだれかと競争するためにするものではありませんから……。受験の場合，そういってばかりはいられないんですけど。がんばってくださいね。

◆そろそろ就職活動をしなければならない時期なのだが，どうもその気にならない。もし霞を食べて生きていけるのなら，就職なんてしなくても平気なのにな。　芳賀　光(20)宮城県

じゃあ，第一歩として山ごもりですか？

◆友人から車を買った。ナンバー変更の手続きをしたら，なんと「68-68」。まさに僕のためにあるようなナンバーでちょっと嬉しかった。で，つい最近，400ccのバイクを買った。これのナンバーは「98-98」だった。というのはウソである。　仁木　直樹(20)北海道

車のナンバーってすべての組み合わせが使われているんでしょうか？　もしそうなら，42-19(死に行く)なんて，ちょっと……。

◆アルシンドの下手な日本語の一方で，ハリソン・フォードにカタカナ英語で「ビール」といわせる○リンのCMはどうかと思ってしまう今日この頃。　喜多　清高(24)兵庫県

ハリソン・フォードも最初は英語で「BEER」と発音していたのが，飛行機内で菊池桃子扮するスチュワーデスに「ビール」といわされていたような気がします。ということは，これについては菊池桃子が悪いということで……。あれ。

◆モデムを購入して，電話代が3倍になった。助けて……。　菊池　真斗(21)北海道

電話代を助けるために，編集部でモデムをもらってあげましょう。

◆後輩(女の子ね)を誘って食事に行った。食べている最中に，「先輩と食べていると食欲が湧いてくるんです」といわれた。別の人からも同じようなことをいわれたことがあるのだけれど，いったいどういう意味なんだろう。

児玉　茂昭(23)京都府

よっぽどおいしそうに食べるんでしょうね。だってそういう人と一緒に食事をすると，いつもより食べてしまいませんか。

◆ゲーム関連の店に限らず，ゲームとはおよそ関係があるとは思えないようなところでも，3DOを見かけるようになった。3DOの目標の100万台の内訳ってもしかしたら，店頭デモがほとんどだったりして……。小島　修(22)神奈川県

5月に発売された3DO関係の某雑誌ではREALが手作りされている写真が載っていました。もし全部を手作りするとしたら，100万台作るだけでもいつになるやら……。

◆ピアノを習い始めます。先生の話では，ピアノは打楽器ではないので音を引き出すように，手前に引くように手に重心かけて弾くのだそうです。確かに，普通に弾くと「コツン」というクリック音のようなものがするのです。難しそうですがいまからとても楽しみです。う〜ん，アップライトでいいからピアノがほしいな。

瀬尾　達人(21)埼玉県

21歳での手習いとはなかなか大変ですが，早くうまくなるといいですね。

◆ウチのディスプレイは，ときどきピーピー鳴いてうるさい。なぜだろう。

鏡味　剛史(19)愛知県

最近餌をあげてないんじゃないですか？　腹一杯，電気を食べさせてみましょう。壊れても責任はとれませんが……。

◆チョコボールにはまってしまった……5つ買って銀が4つあった。あと1つ(笑)。

飯田　英貴(17)茨城県

残り1つが苦難の道か……。

◆受験勉強本格化の時期になりつつある近頃。都会のほうでは相当やっているんだろうな，などと思っている自分が情けない……。話は変わるが，これが載ると掲載4回目になる。なんか俺ってラッキーなんて思ったが，載るたびに母がいう「なんかもらえるの」。このひと言がツライ……。　長谷川　祐之(17)新潟県

次は「ショートプロ」か「LIVE in」で掲載を目指しますか。

◆研究室にいりびたっている。夜が遅いので疲れがたまっている。ふー，明日は土曜日なのに集中講義……夜は研究室でラーメンを食べに行くとか。いいのだろうか，こんな生活で……逃げているような。どうしようもないか！　6月は教育実習……がんばってこよう。しかし，女子クラスの担当……大変だろうな。

中村　学(21)福岡県

すでに教育実習は終えられていると思いますが，そのときは大変でも，いま頃はいい思い出になっていることでしょうね。

◆NetHackは遺言級のゲームです。ROGUEをやったことのある人ならハマッて抜けられなくなることうけあいです。おまけに一度始めると数時間は軽くふっとびます。あー，眠いよー。

石田　伯仁(21)神奈川県

体に鞭打って起きているか，甘い眠りの世界へと身を投じるか，選ぶのはあなたです。眠りにおちたときが，身に危険の及ばないところであることを祈っています。

◆糖尿病と血圧には，気をつけましょう。

関森　一紀(25)神奈川県

ご心配いただきありがとうございます。

◆X68000ACE-HDを使って5年目，ついにX68030を買ってしまった。友人M君にはまだ秘密。このハガキをなるべく目立たないところに載せてください。彼がどのくらい，Oh!Xを読んでるか，これでわかります。

小笠原　義勝(33)神奈川県

このあたりでいかがでしょうか？

◆ついに貯金が底をついた。バイトも決まらないし，来月の駐車場代はどうするんだよ〜！

小山　優一(20)東京都

なにかを売ってとりあえず凌ぐ。さて，ここでなにを売るか？　車を売れば駐車場代まで……(冗談)。

◆港の草地にポータブルチェアをすえ，サングラスをかけ，はだしになって，kissourを手元に，お気に入りの音楽を聞きながら，本を読む。真っ青な空の下，遠くに見える白い雲，行き交う漁船，少し先には釣りを楽しむ老人，足元には小指のつめよりも小さなうすむらさきの花がそよ風にゆれる。最近の私の週末の過ごし方です。気持ちいいよ。　大杉　玲(23)静岡県

◀武田　正道　兵庫県
いかにも夏っていう元気なイラストですね。これを見ていると，海に行きたくなってしまいました。あと，雲の形が……ですね。わかりにくいかな。

◀板橋　芳則　福島県
タイトルは「台風来襲!!」だそうです。でも，この女性のんびりと海なんか眺めていていいのでしょうか。そのうち高波に呑まれて……(冗談)。

どこか別の雑誌に送られてきたハガキかと思ってしまいました。だって，あまりに世界が違うものですから……。

◆現在，5月26日午後1時13分。授業中である。昼食後でとても眠い。講師の話が子守歌のようだ。ウ〜，ウ〜，眠い。日本のマスメディアについての話が続いている。あっ！　い，いかん，講師がこちらに歩いてくる。私の前2メートルのところ，あっ！　私の前で止まった。こちらに向けて冷たい視線が…。上を向くことができない。講師の手が私の肩に……。い，いかん!!　あ〜っ！(ウソ)　加藤　一八(20)神奈川県

今回はウソでも来月には……。

◆大学での授業，友人のノートを写していたら先生に見つかって「君はもうこなくていい」といわれた。……ガーン。皆さん気をつけましょう。　安藤　広明(20)千葉県

ウソではないみたいですね。まあ，あきらめずに出席してみましょう。試験の頃にはすでに忘れているかもしれません。ただ，名前をチェックされていたら……。

◆友人が大麻が1kgいくらという話をしていたので，聞いてみたら，タイ米の話だった。でも，タイ米と大麻って，響きが似ているな〜。

西尾　昌人(20)愛知県

タイマイ（玳瑁）って1mくらいの大きさで，甲羅がべっこうの原料になるウミガメのことですよね。あれっ，失礼しました。

◆6月号にパチンコのプラスでハードディスクが買えるかも，とのメッセージがありましたが，私はその逆でゴールデンウィーク中にX68030がフルセットで買えるくらいマイナスになってしまいました。　中川　敏彦(22)広島県

じゃあ，夏休みにはもう1回負けた気でX68030をフルセットで買うっていうのはどうですか？

◆僕がバイトしているゲーセンには，小さな女の子向けに，セーラームーンの人形ばかりを集めたUFOキャッチャーがあるが，実際にやっているのは，小太りのお兄さんたちばかりだ。

村上　元章(19)静岡県

小太りのお兄さんにはきっと家で小さな子供が待っているんでしょうね(本気3％)。

◆いつのまにやら，剣道部の部長になってしまった。やばい，このままではX68000をさわる暇がなくなってしまう。「パソコン部兼剣道部」なんていう部活があると便利でいいな。

森谷　好雄(16)北海道

なければ作るということで，さっそく学校に申請してみましょう。やっぱり文武両道ですよね。あれ。

◆近くのパソコン店が広くなったのはいいけれどX68000が姿を消してしまいました。しかも，ソフトまでどこへやら。あ，こんなところに「ぷよぷよ」が！　1個。……言葉を失ってしまいました。ところが，そんななかで「Z-MUSIC ver.2.0」が10冊近くあったりします。というわけで，まだ入手していないという方はぜひ，沖縄にお越しください。　藤原　彰人(24)岡山県

まだ入手できないというハガキを見かけますが，そんなわけでこの夏は沖縄ですね。決してどこかの航空会社の回し者ではありませんから。それにしてもこの住所は？

◆やべーぞ，中間試験。成績だれかくれー！

戸村　俊輔(15)千葉県

昔の赤点でよろしければ……。

◆プログラムを組んでいる皆様，プリンタでリストアウトしてますよね。どうしてますか？使用済みのリスト用紙。メモ用紙，なるほど。私のお勧めは，文庫本のブックカバー。リスト中の一番かっこいい(見栄えのする)ページを外側にして使用します。なかなかいいですよ。エコロジーでしょ。だから，本屋では「カバーはいいですから」っていうんです。そーすると，レジのお姉さんが尊敬のまなざしで……(ウソ)。

永井　邦彦(24)愛知県

けっこういけると思います。やっぱり，そのときのリストはアセンブラが一番かっこいい？

◆浪人中の皆さん。大学に入っちゃえば浪人してたかどーかなんてわざわざ聞かない限りわかりませんよ！　安心しましょう。僕？　現役ですよ。

小林　典弘(19)静岡県

いやみにしか聞こえないような気がしますが，確かに聞かない限りわかりません。しかし，一度聞いたら卒業まで忘れませんでしたけど。私？　浪人ですよ。

◆4月からひとり暮らしを始めて，もう6kgもやせてしまいました。このままではヤバイ，と思いつつも今月は「ストリートファイターⅡダッシュ」を買ったので食費を削らないと……。

福嶋　哲也(18)大阪府

そろそろ10kgの大台に突入でしょうか。そうしたら，ダイエット本でも出版するといいかも……。

◆大学生になってからというもの，おなかがすいて目が回るという現象をしばしば体験している……。

冨田　昌胤(20)山口県

社会人になってもおなかがすいて目が回る私はいったい……。

◆公務員試験の倍率が高くて大変です。

生田　淳一(22)京都府

まさに世相を反映しているって感じですね。

◆「ストリートファイターⅡダッシュ」特別編に続いて「餓狼伝説2」特別編もぜひやってほしいです。ところで「キャンセル」ってなんですか？　どうやるの？　まったく知識がない僕としてはそこんところもよく説明してほしいです。

古川　隆(16)大阪府

入力したコマンドの動作(グラフィック)が完全に終わる前に，次のコマンドの動作(グラフィック)が始まることをいいます。やる方法は技のコマンドを素早く入力することです。もちろんすべての技に関してできるわけではありませんよ。念のため。

◀岩瀬　貴代美　福岡県
福岡といえば福岡ドームですね。なんといっても今年はダイエーが強いし。仕事の帰りになんか見にいけるといいですね。がんばれダイエー！

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

売ります

★ローランドの音源モジュール「SC-55」とシステムサコムのMIDIボード「SX-68MⅡ」をセットで35,000円(送料込み)で売ります。箱，説明書，付属品すべてあります。バラ売りは不可です。連絡は往復ハガキでお願いします。〒272　千葉県市川市大野町1-82-3　石原　大助(18)

★アイ・オー・データ機器の2Mバイト増設RAMボード「PIO-6BE2-2ME」を14,000円，SCSIボード「CZ-6BSI」を11,000円，サイバースティック「CZ-8NJ2」を9,000円で売ります。送料込みで，箱，説明書，付属品すべてあります。連絡は往復ハガキでお願いします。〒569　大阪府高槻市芥川町2-3-16　木下　勝文(22)

★アイ・オー・データ機器の4Mバイト増設RAMボード「PIO-6BE4-4ME」を30,000円(送料込み)で売ります。連絡は往復ハガキでお願いします。〒245　神奈川県横浜市泉区和泉町6239-3　蓑和田　啓太(30)

★RGBシステムチューナー「CZ-6TU」を15,000円(送料込み)で売ります。テレビコントロールケーブルつき。ただし，箱はありません。連絡は往復ハガキでお願いします。〒615　奈良県奈良市西京区桂艮町25-29　市田　治男(66)

★SCSIボード「CZ-6BSI」を15,000円(送料込み)で売ります。箱，説明書，付属品はすべてあります。連絡は往復ハガキでお願いします。〒064　北海道札幌市中央区南13条西22-1-10　小林慎(19)

★モニタテレビ「CZ-614D」を50,000円，カラー漢字プリンタ「CZ-8PC5」を30,000円，カラーイメージユニット「CZ-6VT1-BK」を25,000円，アイ・オー・データ機器の4Mバイト増設RAMボード「PIO-6BE4-4ME」を25,000円で売ります。また，カラーイメージスキャナ「JX-220X」とアイ・オー・データ機器のスキャナ用パラレルボード「SH-6BN1」をセットで70,000円で売ります。すべて箱つきです。スキャナやモニタは，なるべく取りにくることができる方を優先します。連絡は往復ハガキでお願いします。〒320　栃木県宇都宮市宝木町1-2584-7　コーポタカラギ202号　大野　周一(22)

買います

★カラーイメージユニット「CZ-6VT1」を送料こちらもちで，30,000円で買います。説明書と付属品があれば箱はなくてもかまいません。連絡は官製ハガキでお願いします。〒958　新潟県村上市飯野桜ケ丘3-20　富樫　春男(45)

★シャープの拡張I/Oボックス「CZ-6EB1」を45,000円前後で買います。完動品であれば，箱，説明書はなくてもかまいません。連絡は往復ハガキでお願いします。〒930　富山県富山市永楽町14-9　多賀　陽介(24)

★シャープかシステムサコムのSCSIボードを15,000円程度，同メーカーのMIDIボードを10,000円程度で買います。また，RGBシステムチューナー「CZ-6TU」を15,000円程度で，ジャストの拡張SIMMメモリボード「ERIOS」を10,000円程度で買います。いずれも説明書と付属品ありでお願いします。価格は送料込みの値段です。連絡は希望価格を書いて往復ハガキでお願いします。〒206　東京都稲城市矢野口1161-223　長石裕行(24)

バックナンバー

★Oh!Xの1993年7～9月号をセットで3,000円(送料別)にて買います。汚れ，ハガキ以外の切り抜きのないものをお願いします。連絡は往復ハガキでお願いします。〒182　東京都調布市国領町4-35-2-720　井芹　洋之(16)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今月は6月号の内容に関するレポートです。

●ドラゴンだ。素晴らしい。毎回このくらいパワフルな特集を期待しちゃうよ。つなげてみたいけどできるかどうかわからないもの，X680x0用だけどなんだかよくわからないもの，その他もろもろを実際に試用してくれるというのは，大変ありがたいです。ラインナップも注目の040turboやマルチスキャンモニタから謎のPPIボード，無停電電源装置まで充実していました。

X68000の周辺機器は「他機種の安い品物を（無理やり）つなげる」というのが基本的なスタンスのようなので，次回はぜひ「PPIボードでIDE-IF」とか「98用3.5インチFDDをつなぐ」とか「とにかく変なものをつなぐ」などなど，ローテクな特集もやっていただきたいです（瀧氏が死にそう）。

石田　伯仁（21）X68030，MZ-731，PC-8801mkMR，PC-9801VMII，PC-E200　神奈川県

●周辺機器はすごいものだと思います。もはや，ハードディスクなしのパソコン環境など考えられませんし，増設メモリもないと苦しいです。これらのない国民機やX68000は快適ではありません。AMIGA2000（現在はAMIGA4000が主流だけれど）は単体のグラフィックや音源の性能ではX68000にはかないません（と思う）。しかし，世界中の豊富な周辺機器を接続することにより，最強のDTVマシンやCGマシン，DTMマシンに変身します。X68000にもそのような周辺機器の登場を望みます。

松永　孝治（24）　X1turbo model30，MZ-80C，PC-9801N，AMIGA1200　鳥取県

●特集ですが，「メガディスプレイへの道」がよかったです。以前にSX-WINDOWで－Gオプションを使って1024×848ドットを試したことがありますが，とても我慢できるものではありませんでした。でも，少しでも広い画面にしたかったので，CRT960を使用していましたが，この記事を読んだことで「少しでも広く見やすい画面」を目標に，レジスタ値の計算に励んでいます。

森崎　剛（22）　X68000 XVI，PC-9801RX21，FM TOWNSII HR20　広島県

●タブレットには大変興味があるのですが，なにぶん「MATIR」と「ハイパーピクセルワークス」でしか対応していないのと，値段が高価なのが少々問題でした。マウスに比べると，より「自由に絵が描ける入力デバイス」ではありますが，対応ソフトが少ないのがいちばんの問題です。付属のマウスと同様に使用できるデバイスドライバなどがあったらいいなと思います。

八亀桂一（19）　X68000 PRO　神奈川県

●「これだけ出来のええゲームをけなす奴ぁおらへんやろ」と思っていただけに，八重垣氏の「ジオグラフシール」のレビューはショックでした。コンシューマやPC-9801のポリゴンものを見てきた私には，とんでもないソフトだったからです。最初は「ここまでボロクソいうか」と思ったのですが，「世の中いろんな人がいるのだ」と考えることにしました。

ただ，いわせてもらうなら「俺はこう思う。おめえらもそう思うだろ。そーだろ。そーだっていえよ」っぽい文章だったように思います。意見を述べるなとはいいません。でも，その意見を他人に押し付けるような言い回しは危険な気がします。

中矢　史朗（23）　X68000 ACE-HD，X68030，PC-386P　愛媛県

●「ローテク工作実験室」で作成した連動コンセントは，あると本当に便利ですよね。私も以前，電脳倶楽部に回路が載ったときに作って重宝しています。電脳倶楽部の回路には，ダイオードが入っていなかったので，記事を読んで慌ててつけ足しました。まぁ，いままで数カ月間無事だったのですから慌てることもなかったのですが，念のため……。

中村　健（24）　X68000 ACE-HD，PC-386GS，MSX2＋　埼玉県

●「アンケート分析大会」ですが，所有機の移り変わりは当然として，メインメモリが2Mバイトがこんなに多いとは！　私のように，ほとんど電脳倶楽部READERになってしまっているマシンが多いのであろうか（初代機でメモリ増設は辛い）。カラープリンタの普及具合，メーカーサポートで買っている人が少ないなど，X68000らしい。現状をよく表しているような気がする。

内藤　陽一（27）　X68000 ACE，PC-98NS/E　東京都

●「F-Calc for x68k」ですが，この時期にとてもコンパクトな表計算ソフトを出すのは立派です。多くのソフトがあれもこれもと余計な機能（失礼）を増やしていって，気がつけばHDD必須，処理は重たい，使えない！　といったものになっています。そのなかで，基本的なところはしっかり押さえつつコンパクト！というのはよいと思います。さらにソースも入手できるとあれば，パワーユーザーがカスタム化して使いやすい（汎用性は低下するかもしれないが）ものになっていくのではないかと，パワーユーザーでない私は甘い希望をもっています。

野原　賢次（33）　X68000 ACE-HD　埼玉県

ごめんなさいのコーナー

7月号　LIVE in '94

P.74　異世界に光る3本の剣

リスト1の90行目に印刷されていない部分がありました。行の最後に「'」を追加してください。ご迷惑をおかけしたことをお詫びいたします。

7月号　付録ディスク「GENIE」

同ディスクの実行時には，フリーソフトウェアのTwentyOneの常駐を解除しておいてください。エラーの原因になります。それが嫌な人は¥GENIE¥binのディレクトリ内を，

MECHANICS.K → MECHANIC.K

に名前変更してください。

7月号　STUDIO X

P.162　神谷　正樹さんのコメントが印刷工程のミスで抜け落ちてしまいました。どうも申し訳ありませんでした。同コメントは，

ただいま消灯3分前，ゴキブリに逃げられた。なんだか寝るのがコワイ！

でした。

7月号　ごめんなさいのコーナー

P.164　お騒がせの「PUSH BON!」36面ですが，解けたとのお便りやお電話をいただきました。手順を解説してくださった，鎌田さん，高橋さん，河田さんほかの皆さんどうもありがとうございました。現在，42手での手順までわかっていますが，39手で解けたとの報告もいただいています。まだ解けていないという人はがんばってみてください。

バグに関するお問い合わせは
☎03(5642)8182(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

広がる可能性は あなた次第

▼パソコンでグラフィックをするといってもいろいろあります。たとえば,「MATIER」や「Z's-STAFF」などのグラフィックツールでマウスなどのデバイスを使って自分の好きな絵を描くこと。これももちろんパソコングラフィックですが,自分の好きな絵を描くだけなら,鉛筆と紙があれば描けます。では,パソコンでないとできないグラフィックというとどんなことでしょう。

自分で描いた絵に何度も手を入れたり,一部を描き直したり,手描きではなかなか面倒なことです。でもこれくらいなら,苦労すればできるかもしれません。では,フレアやランダムフラクタルなどよばれる特殊なエフェクトを自分の手で施すことができるでしょうか? まず無理でしょう。でもパソコンなら簡単にエフェクトをかけることができます。

いろいろな特殊効果を使って,コンピュータグラフィックの広がる可能性をあなたも試してみませんか。そして,あなたのマシンにも美しいグラフィックを描きましょう。

▼先月号で決定した愛読者年間モニタですが,追加発表を行います。といっても,石田伯仁(神奈川県)さんひとりだけなんですが……。合わせて14名の方々の手元にはすでにレポート提出用の質問が届いていると思いますので,よろしくお願いします。

▼「ファイル共有の実験と実践」「SX-BASIC公開デバッグ」はページの都合により,また「X68000マシン語プログラミング」「システムX探偵事務所」は著者多忙のためお休みとさせていただきました。楽しみにされていた方には,お詫び申し上げます。

▼あと,ごめんなさいの追加です。付録ディスクでオマケとして収録されていた「TCH2AMI.X」なのですが,一部のマシン(おそらくX68030)で動きません。再コンパイルをすれば動くと思われますが,当方ではソースを所持していないため,対処しかねます。残念ですが,オマケはオマケということで……。どうもすみませんでした。

投稿応募要領

●原稿には,住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。

●プログラムを投稿される方は,詳しい内容の説明,利用法,できればフローチャート,変数表,メモリマップ(マシン語の場合)に,参考文献を明記し,プログラムをセーブしたフロッピーディスクを添えてお送りください。また,掲載にあたっては,編集上の都合により加筆修正させていただくことがありますのでご了承ください。

●ハードの製作などを投稿される方は,詳しい内容の説明のほかに回路図,部品表,できれば実体配線図も添えてください。編集室で検討のうえ,製作したハードが必要な場合はご連絡いたします。

●投稿者のモラルとして,他誌との二重投稿,他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒103 東京都中央区日本橋浜町3-42-3
ソフトバンク出版部
Oh!X「㋢㋮㊔」係

SHIFT・BREAK

▶今年の夏はバイクで北海道の実家へ帰ろうと思う。3度の飯をキチンと食べ,ゴミの散乱していない部屋で朝日の昇る頃,自室のベッドの上で目覚める生活をするのだ。1日1食,無数のゴミの山のなかで夕日の沈む頃,ゴミ捨て場から拾ってきた布団の上で寝起きする生活とは,しばしお別れ。ああ,採れたてのトマトが食べられる。(H)

▶パソコンのキーボードは安っぽい。先日原稿書き用のPC-286VFを買い替えようと秋葉原に出かけたが,性能はともかくどれもキーボードがちゃちいのに嫌気がさして,結局アクセラレータでお茶を濁すことにしてしまった。ナナオがディスプレイ界のフェラーリといわれるように,キーボードにも頂点を極めるメーカーが出てこないかね。ホント。(E.K)

▶SX-BASICはゲラまで出たけど,本当にページの都合で来月送り。楽しみにしていた方々ごめんなさい。アクセラレータはデータキャッシュを使えるようになったが,締め切りの都合で来月送り。スタンフォードテストで約2倍。非同期化クロックはまだまだ不安定。やっぱ,3倍はほしいよなぁ。ということろで,また来月。あぁ,1/5の幸せ。(石)

▶ああっ,むしょーに腹がたつ! 何かネタを思いついたら書き留めておこうとせっかくネタ記録帳を作っておいたのに……。でもって,すごいネタを思いついたはずだったのに。いくら眠かったとはいえ,たったひと言しか書いてないなんて。

「なまこのまるかじり」

何が書きたかったのかさっぱりわからん……。(で)

▶走り込みのかいあって,「デイトナUSA」の中級をようやく完走した(軟弱だがオートマのタイムアタックモード,視点は0)。私にはタイム設定が少々きつすぎるが,ドリフトを理解してからは走ることが楽しくなってきた。やっぱりアナログの操作系と60fpsは譲れないよ。(というわけで次世代ゲーム機のパッドにそこはかとない不安を感じているA.T.)

▶新しく椅子を買った。座ると仕事をせざるを得なくなるようなモノって視点で探したら,ドイツ製にいきついた。座面は丸く,どの角度で座ってもしっくりくるようになっているし,堅めで腰が沈まず,深く座ると,自動的に背筋が伸びる設計になっているという,非常にドイツな代物であった。しかし,輸入物の椅子はなんでこんなに高いのだ。(K)

▶長い間療養生活を続けているせいか,やらねばならないことを先に延ばしている。ふと気がつくとムーンプリズムパワーBOXの締め切りが近い。あのLDは3枚しかもってなかったので,急いで残りの9枚を買いあさる。そして毎日1枚ずつ観賞するのを日課にしている。昔と比べると,みんなのいうように,Sはパワーダウンの感がいなめないと思う。(KO)

▶う~ん。口の中が痛い。口内炎になってしまったようだ。そりゃ,こんな不規則な生活を送っていれば,当たり前(?)。ある雑誌で面白い薬を見つけ,試してみる。シールのようなものを患部に直接貼るのだが,結構いけるかもしれない。商品名は「口内炎パッチ大正A」。現在まだ治っていないのだが,ただのビタミン不足かそれとも内臓……。(高)

▶ある平日の昼下がり。街なかをふらふらしていたら,変なおじさんが寄ってきた。「OL?」「違います」(ひらひらのミニスカートのOLなんていないぞ,きっと)。「学生?」(そんなトシには見えないでしょ)。「フリーター?」「……」。この選択肢なら,やはりOLというべきだったのか。でも,生活態度は最後のがいちばん近いかも。うーん困った。(ふ)

▶……オリジナルステージデータがこない。ま,それはともかく,ちょくちょく投稿されてくるゲームを遊ぶのだが,これが結構面白い。一級品といえるものはないけど,ゲーム作りを楽しんでいる様子が作品に込められていて,ついつい遊び惚けてしまう魅力がある。こういった作品たちを,もっともっと読者のみんなに届けられると面白いかもね。(J)

▶夏も近づく8月10日……。今年はちと早いよなあ。気がつけば,再来月は付録ディスクの予定。ディスクの名前も特別定価もまだ未定だ。今度こそあまり詰め込まないようにしよう,と思いつつ,一応,「小物をたくさん」ということでいろいろ手配してはいるが……。個人的には100Kバイトくらい余った付録ディスクってのがいいなあ。(U)

▶Oh!Xといえば,数あるパソコン雑誌のなかでも,伝統のユーザー気質を守る名門誌(当社比)。それも熱心な読者に支えられてのことですが,それでも編集部の台所は火の車で社内での立場も危うい状態です。そこで,申しわけないのですが,次号より定価を680円に値上げさせていただくことになりました。どうかご理解のほどお願いいたします。(T)

microOdyssey

「……はっ，いかん。もう朝……。今日はテストだというのに………。」

このくらいならまだいい。

「……俺が道の上に寝ている。俺はなんで空を飛んでいるんだ。ひょっとして死んだ……？」なんてこともあるかもしれない。眠ってしまうことによって。

眠気は突然やってくるような気がする。では，眠気とはなんだろうか。人間が生きていくうえで必要不可欠なものだ。とすれば，食欲などと同じ生理的欲求である。実際には眠気を感じないときがある。スポーツをしたり，人と話をしたりして刺激がある場合だ。

これは空腹のときに似ている。つまり，空腹が進行しているときも，いつも空腹を感じるわけではない。なにかをしていて，空腹を忘れるときがあるだろう。ただ，そのときも空腹であることには変わりない。眠気の場合も同様で，刺激がある場合には忘れているが，そのときも生理学的には眠いということだ。そして，眠気を解消するには眠るしかない。

ところで，皆さんは小さい頃にどのくらいの睡眠時間をとっていただろうか。私は8時間以上の睡眠をとっていたが，皆さんも似たようなものだと思う。では，それが高校の頃ではどのくらいになっただろうか。私の場合，減ってしまった。

ある実験によれば，10歳のときよりも16～18歳のときに長い睡眠時間を必要とするという結果が出ているそうだ。ということは，高校の頃の私の睡眠時間は足りなかったらしい。そして，足りない睡眠時間は，まるで負債を背負うかのように蓄積されていくらしいのだ。授業中の居眠りで消化していたという気もする。皆さんも睡眠時間が減少していれば，同じように蓄積されているはずだ。もちろん，個人差はあるのだが。

そして，必要な睡眠時間は歳をとっても劇的に減少するわけではないらしい。朝夕の通勤電車を見てみるといいだろう。座っている会社員やOLの方が眠っている。皆，負債を支払っているのである。

ある程度規則的な生活を送っている人でさえこの状態である。社会にはいつ睡眠をとるか，自分でコントロールできない人が多い。たとえば明日が締め切りなのに原稿を上げていないライター，看護婦などの交代勤務者，朝までに入稿しないといけない編集者，夜間に働くタクシーやトラックの運転手などなど。

そんな人たちが，起きているときに適切な活動を行うための助けとなる手段は，眠気が襲ってきたときに腹痛や頭痛などの痛みが起こったときと同様に重大なことと受け止め，仮眠をとることだろう。とくに，ほんの一瞬の眠りが生命の危険や重大な事故を及ぼすような状況では当然である。

世界のいろいろな場所で午後の昼寝(シエスタ)が大切にされている。ここらで日本も仮眠(昼寝)の時間を取り入れてくれないだろうか(もしくは会社でも可)。睡眠に関する研究者もいっている。うたた寝をしている人は，理性的で思慮深く素敵な人とみなされるべきである，と。別に私が仮眠をとっている言い訳をしているわけでは……。（高）

1994年9月号8月18日(木)発売

特集　SX-WINDOWで創造する

・新しいシャーペンの実力を探る
・シャーペンの外部ファイルを作る
・そのほかSX-WINDOW ver.3.1の活用法

試用レポート
X68030改：33MHz版/H.A.R.P
GAME REVIEW
餓狼伝説SPECIAL

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(3233)3312
	〃	書泉ブックマートBI	03(3294)0011
	〃	書泉グランデ5F	03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(3257)2660
	八重洲	八重洲ブックセンター3F	03(3281)1811
	新宿	紀伊国屋書店本店	03(3354)0131
	高田馬場	未来堂書店	03(3209)0656
	渋谷	大盛堂書店	03(3463)0511
	池袋	旭屋書店池袋店	03(3986)0311
	八王子	くまざわ書店八王子本店	0426(25)1201
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	043(224)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある「新規」「継続」のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

基本的に，定期購読に関することは販売局で一括して行っています。住所変更など問題が生じた場合は，Oh!X編集部ではなくソフトバンク販売局へお問い合わせください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

Oh!X 8月号
■1994年8月1日発行　定価600円(本体583円)
■発行人　橋本五郎
■編集人　稲葉俊夫
■発売元　ソフトバンク株式会社
■出版事業部　〒103 東京都中央区日本橋浜町3-42-3
Oh!X編集部 ☎03(5642)8122
販売局 ☎03(5642)8100 FAX 03(5641)3424
広告局 ☎03(5642)8111
■印刷　凸版印刷株式会社

ソフトバンクの新刊・近刊

DOS/Vスーパードライバーズ32

●小山隆史 著
●定価9,800円

DOS/Vの日本語環境をV-Text化（高解像度表示やマルチフォント表示）するためのディスプレイ&フォントドライバ集。今回のバージョンは386のプロテクトモードで動作し、32ビットグラフィックチップの大半をサポート。最近主流のVLバスなどに対応しています。

IBM-PC/AT互換機ガイドブック3 DOS6/Vコマンドリファレンス

●DVS Lab. 著
●定価1,800円

PC DOS J6.1/V, PC DOS J6.3/V, MS-DOS 6.2/Vに対応した、DOS/Vのコマンド解説書。各コマンドの概要と指定可能なオプションの解説とともに、主要なコマンドには使用例や参照コマンド名を併記するなど、ユーザの便宜を画りました。

VZ エディタ Ver1.6強化書

●上村郁夫 著
●定価2,700円

エディターも種々あれど、その使い勝手の面、また価格に着目すれば、現在、VZエディタ以外に選択肢を拡げる必要はありません。そんなコストパフォーマンスに格段の優位を誇るVZ エディタの最新バージョン1.6の先駆性ををわかりやすく解説しました。

一太郎・花子・三四郎 らくらく使いまわし術

●金田知之 著
●定価1,800円

一太郎Ver.5、花子Ver.3、三四郎Ver1.1を、ジャストウィンドウを介して自在に使いこなすためのノウハウを詳述した、ユーザ待望の解説書。今まで気付かなかった便利な使い方が満載です。

ネットワーカーへの道

●大原まり子 著
●定価1,500円

SF作家大原まり子の『Oh!PC』連載エッセー「怒涛のビギナーズライブラリ」とパソコン通信関連のエッセーをまとめて書籍化したものです。98を使いはじめ、通信世界にずぶずぶと身体を埋めてゆく、涙ぐましくも楽しい日々がいきいきと描かれています。

X680x0 TeX

●吉野智興・川本琢二・山崎岳志・実森仁志 共著
●定価9,800円

はじめてのX680x0版TeXの解説書。はじめてTeXを使う人には簡単インストーラーを付けて基本的な使い方を、すでにTeXを使い込んでいる人にはカスタマイズにしかたや、数学記号などの表記に優れたAMSTeX、楽譜が書けるMUSIC-TeXなどのサンプルや、縦書きマクロ(アスキー、インプレス開発)などの周辺ツールを付けています。

スーパーファミコン オールゲームカタログVol.2 完全保存版

●Theスーパーファミコン編集部特別編集
●定価1,100円

昨年発売された"Vol.1"に、さらに新データを加えたスーパーファミコンソフトのオールゲームカタログ。94年5月31日までに発売されたソフトの紹介、攻略本、評価など全データを掲載しています。あわせて周辺機器の情報もフォローしています。

餓狼伝説SPECIAL スーパーガイド

●Theスーパーファミコン編集部 編
●予価720円

ゲームセンターで大人気の、SNKより発売された格闘アクションゲーム「餓狼伝説SPECIAL」の完全攻略ガイド。登場キャラ16人のプロフィールと通常技の基本操作、必殺技コマンドの操作法などを前半で紹介、後半では各キャラクターを使用した場合の〈対戦攻略〉を中心に紹介します。必殺技の使いどころや連続技など、対人プレイに関するテクニックを紹介します。

SOFTBANK MOOK

ザウルス出現！MarkⅢ

●定価1,500円

ザウルスは進化した、外出先でさらに機動力を発揮するために。Newザウルス(PI-4000)の新装備　インクワープロとは？/FAX送信の簡易設定など、ザウルスを携帯情報端末として高度利用するための提案を満載。

CONTENTS

1.これがNewザウルスだ
2.外出先でザウルスを使う
3.ザウルス活用法
4.ザウルスの基本機能
5.ザウルスの文字入力
6.ザウルスとパソコンを接続する
7.アシスト機能
8.インクワープロ/FAX送信・アシスト機能全公開

HP Dynamism Vol.2

HP-UXアドバンストユーザーへの道

●定価1,500円

HP-UXによるシステム構築から、運用のコツ、プログラミング例まで、多彩で豊富な情報を提供します。また各種フリーソフトウェアのHP-UXへの移植、Perlによるプログラミング、シェルの使い方などを詳しく解説し、新機種712やCOSEのデスクトップ環境なども紹介します。

CONTENTS

PART1　再生するUNIX～脱皮と継承
PART2　HP-UXをめぐる新しい動き
PART3　もっと使いこなそう、HP-UX
PART4　ちょっとハイブローなHP-UXの楽しみ
PART5　ビジネスを支えるHP-UX

定価は税込です
お近くの書店でお求めください

SOFT BANK ソフトバンク株式会社/出版事業部

“夏・ツクモ・ザ・バーゲン”

(半期に一度の決算セール!!)

ツクモラジオCM.好評 ON AIR中!!
東京 ニッポン放送「伊集院光のoh!デカナイト」内
(月)~(木) PM10:30ごろ
名古屋 東海ラジオ 「大内ひろのしんの今夜もやっぱりFUNKYパジャマ」内
(月)~(金) PM11:00ごろ
札幌 FMノースウェーブ「SUPER CHOICE」
(月)~(金) PM7:30ごろ
ツクモイメージガール越智静香さんが茶目っ気たっぷりにパソコンをおねだり。
月末には特価品をズバリ!ツクモ特価でお知らせします!!
~これを聴いたら、ツクモに来ないではいられません~

TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKUMO

お申し込みは今すぐ! 受注専門フリーダイヤル **0120-377-999**

~ツクモなら安いし、品揃えも豊富だし、スタッフだって…ツクモに御来店下さい!~

X680x0シリーズ本体

好評につき引き続き **71%OFF!**

CZ-674C-H(X68000 CompactXVI)…
超特価¥84,800

TS-XFDCAを使えば、縦置き5インチモデルX68000シリーズ(PROシリーズを除く)を外部ドライブとして使用可能!

※モニタは別売です

お勧めのセット

CZ-674C-H… ¥298,000
CZ-607D-BK… ¥99,800
RGBケーブル…… サービス
ツクモ特価¥148,000

X68030のお勧めの組み合わせ!!

CZ-500C-B… ¥398,000
290MBハードディスク サービス
ツクモ特価¥296,000

電子文具

これぞ、パーソナルシステムの決定版!!

PI-4000
定価¥75,000
ツクモ特価¥59,800

PI-4000FX FAXモデムセットモデル
定価¥91,000
ツクモ特価 ¥72,800

満開製作所の商品も取扱中!

X68000 CompactXVI 24MHz改
RED ZONE……… **ツクモ特価¥130,000**
RED ZONE(2DD)……… **ツクモ特価¥135,000**

満開製外付け5インチFDD
MK-FD1……… **ツクモ特価¥ 39,800**
MK-FD1(カラーリングモデル)……… **ツクモ特価¥ 44,800**

X68000/030シリーズ用RAMボード

商品	ツクモ特価	商品	ツクモ特価
SH-6BE1-1ME(CZ-600C専用)	¥10,800	TS-6BE2B(CZ-6BE2A/D用拡張RAM)	¥29,800
PIO-6BE1-AE(ACE/PRO/PRO2シリーズ用)	¥10,800	TS-XM1-4A(拡張スロット用4MB)	¥39,800
PIO-6BE2-2ME(拡張スロット用)	¥22,800	TS-XM1-6A(拡張スロット用6MB)	¥47,800
PIO-6BE4-4ME(拡張スロット用)	¥38,800	TS-XM1-8A(拡張スロット用8MB)	¥55,800
SH-5BE4-8M(X68030シリーズ用)	¥44,800	TS-XM1-10(拡張スロット用10MB)	¥63,800
CZ-6BE2A(XVI専用)	¥42,500	X SIMM10-8M(拡張スロット用8MB)	¥53,800
CZ-6BE2D(CompactXVI専用)	¥29,800	X SIMM10-10M(拡張スロット用10MB)	¥64,800

TS-3XRシリーズ

X680x0用外付けドライブ
・2DD/2HD/2HC/1.44MBフォーマット対応
※2DD/2HC/1.44MBを使用するにはHuman68K Ver.3.0以上が必要
●CompactXVI/68030用ケーブル付

TS-3XR1B 1ドライブ 定価¥33,800………… **ツクモ特価¥26,800**
TS-3XR2B 2ドライブ 定価¥46,800………… **ツクモ特価¥36,800**

ツクモオリジナルシリーズ

ジョイスティックパラレルインターフェイス

TS-JPIFS 定価¥17,800 **新発売!!**

拡張スロットを使用しません。ジョイスティック端子に接続できるパラレルインターフェイスです。
これでスキャナーも高速で取り込みが可能になります。
取り込みソフトェア及びサンプルソースが付属致します。

発売記念特価 ¥14,800

ディスプレイも特別価格にて提供中!

CZ-607D(14型カラーディスプレイテレビ) **ツクモ特価¥ 60,000**
CZ-608D(14型カラーディスプレイ) **ツクモ特価¥ 69,000**
CZ-615D(15型カラーディスプレイテレビ) **ツクモ特価¥132,000**
CZ-621D(21型カラーディスプレイ) **ツクモ特価¥125,000**

プリンター

マッハジェットカラー
MJ-700V2C
(ケーブルセット)

ツクモ特価¥83,800

カラーバブルジェットプリンター

ツクモ特価¥86,800

バブルジェットプリンター
BJ-10VLite(ケーブルセット) **ツクモ特価¥33,800**
BJ-15VPro(ケーブルセット) **ツクモ特価¥43,000**

カラーイメージスキャナー

JX-330X NEW 定価¥178,000

基本解像度(600dpi)、超高速が特長。
ADF・透過原稿対応型カラーイメージスキャナの登場です。
Scanner Tools(画像入力ソフト)付属。

ツクモ特価¥138,000

CZ-8NS1
台数限定 **ツクモ特価¥ 69,800**

【東京】●パソコン本店(各種パソコン・周辺機器)●パソコン本店II(パソコン・ワープロ)●DOS/Vパソコン館(DOS/Vパソコン・下取り)●万世店(総合通信機器)●5号店(ビデオ・ムービー・CS)●ソフト8号店(パソコン&ゲーム用ソフト)●買取センター(ゲーム機・ゲーム機用ソフト買取り)●ニューセンター店(パソコン・中古・下取り・買取り)●ラジオセンター店(ハンディーレシーバー・テレホンパーツ)
【名古屋】●名古屋1号店(パソコン全般)●名古屋2号店(パソコン全般・総合通信機器・ビデオ) 【札幌】●札幌店(パソコン全般・総合通信機器)●DEPOツクモ2番街店(パソコン全般)

秋葉原電気まつり
7月31日(日)まで
★5千円お買い上げ毎に抽選券1枚プレゼント
1等:10万円　107本(各組1本)
2等:5万円　214本(各組2本)
3等:5千円　4,800本(各組39本)

12回払い、7.5%がナント6%に!

クレジット金利がこんなにお安くなりました!
〜月々ムリのないお支払い額で欲しかったパソコンがお手元に!!〜

支払回数(回)	1	3	6	10	12	15	18	20	24	30	36	42	48	54	60
分割払い手数料率(%)	2.5	3.5	4.5	5.5	6	9	11.0	12	12.5	16.5	17.5	22	23	28.5	29.5

TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKU

受付時間(平日)AM10:45〜PM7:30
(日・祝) AM10:15〜PM7:00
木曜定休

『FAX24時間お見積もり受付』
03-3255-4199
お名前.住所.電話番号.FAX番号をご記入の上ご依頼下さい。

ツクモグローバルJCBカード
JCBならではの国内・海外サービスにツクモオリジナルの特典をプラス。ツクモ各店にある入会申込書にてお申し込み下さい。くわしくはグローバル事務局03(3251)9898又は各店へ。
※ジャックス・VISA・セントラル・マスターも取り扱っております。

動画を始めてみませんか?
ビデオ入力ユニット **CZ-6VS1** 標準価格¥178,000

MC68EC020(25MHz)の32BitMPUを搭載し、SCSIインターフェイスを介してパソコンへデータを転送。動画・静止画を簡単に保存出来るアプリケーションソフト「ライブスキャン」を標準装備。1,677万色まで対応し、最大640×480ドットの高解像度で、高速取り込みが可能です。但し680x0シリーズでご使用の場合には6万5千色までの表示となります。

ツクモ特価¥142,000

大容量記憶装置　ハードディスク
240MBハードディスク……ツクモ特価¥39,800〜
290MBハードディスク……ツクモ特価¥46,800〜
350MBハードディスク……ツクモ特価¥49,800〜
520MBハードディスク……ツクモ特価¥74,800〜

MO特選セット
SCSIボードが必要な場合にはセット価格に¥22,000加算となります。

Logitec **LMO-200**(128MB) ¥79,800
プラス SCSIケーブル / MOメディア / ターミネータ
ツクモ特価¥76,800

Logitec **LMO-400**(230MB) ¥158,000
プラス MOメディア / SCSIケーブル / ターミネータ
ツクモ特価¥138,000

Panasonic **LF-3200B**(230MB) ¥168,000
プラス SCSIケーブル / MOメディア / ターミネータ(SW)
ツクモ特価¥138,000

CD-ROMドライブ
CD-ROMドライバーソフト+ケーブル¥　9,200

CD-ROMドライブ(2倍速)
		ツクモ特価
ELECOM	ECD-250(TOSHIBAドライブ)	¥42,800
東芝	XM-4100A(TOSHIBAドライブ)	¥47,800
Logitec	LCD-550-DV(TOSHIBAドライブ)	¥39,800
SONY	CDU-7811(SONYドライブ)	¥45,800
緑電子	CXA-301S(NECドライブ)	¥29,800
PIONEER	DR-U104X(4倍速)	¥69,800

多連装CD-ROMドライブ
PIONEER　ツクモ特価
DRM-602X(6連装2倍速) 台数限定 ¥64,800
DRM-604X(6連装4倍速) <輸入品> ¥135,000

コンピュータアート スーパーグラフィックツール
その1.慣れてしまうとマウスがいらない
DrawingSlate…………¥74,800
Matier Ver2.0…………¥39,800
合計定価¥114,600
ツクモ特価¥89,800

その2.ハイオリティなのにこんなに安い
BJC-600J…………¥120,000
プリンターケーブル……¥4,800
Matier Ver2.0…………¥39,800
合計定価¥164,600
ツクモ特価¥116,000

MIDIコンピュータミュージック特選セット
Roland SC-55mkIIセット
SC-55mkII…………¥69,000
TS-6GM1…………¥39,800
MIDI変換ケーブル…………¥4,000
合計定価¥112,800
ツクモ特価¥79,800

Roland SC-88セット
SC-88…………¥89,800
TS-6GM1…………¥39,800
MIDI変換ケーブル…………¥4,000
合計定価¥133,600
ツクモ特価¥99,000

簡単コンピュータミュージック
Music Card for X680x0
(TS-6GM1)
発売記念キャンペーン価格
¥24,800
・音源を搭載したMIDIボードの登場。これ1枚で手軽にMIDIコンピュータミュージックが楽しめます。GM規格・MT-32・CM-64等の音色配列をサポート。最大同時発音数16。ソフトウェア「Mu-1GSお試し版」付

パソコン通信
モデム
		ツクモ特価
AIWA	PV-AF144V5	¥32,800
OMRON	MD144XT10V	¥33,800
MicroCORE	MC14400FX	¥32,800
Panasonic	TO-703B	¥32,800
SUNTAC	MS144AVF	¥29,800

通信ソフト
		ツクモ特価
SPS	た〜みのる2	¥13,000
SHARP	Communication SX-68K	¥16,800

ソフトウェア
	ツクモ特価
OS-9/X68030 V2.4.5	¥20,000
Technical Tool Kit V.2.4.5	¥16,000
Ultra C & Professional Pack V1.1	¥36,000
X Windows V11.5	¥24,000
SX-WINDOW Ver3.1システムキット(NEW)	¥18,200
SX-WINDOWデスクアクセサリ集	¥11,800
C COMPILER Ver2.1 NEWKIT	¥35,800
Easydraw SX-68K	¥15,800
Easypaint SX-68K	¥10,200
SOUND SX-68K	¥12,600
Communication SX-68K	¥15,800
Matier Ver2.0	¥29,800

	ツクモ特価
CD-ROM Driver	¥4,800
SX-PhotoGallery	¥15,800
DoubleBookin'	¥12,800
SX広辞苑(CD-ROM別)	¥17,800
EGWord SX-68K	¥47,800
SX-WINDOW開発キット	¥31,800
開発キット用ツール集	¥10,200
倉庫番リベンジSX-68K	¥5,400
MUSIC SX-68K	¥30,400
XDTP SX-68K	7月リリース予定
フォントデザインツール書家万流 SX	COMMING SOON
Super BUSINESS	NOW WAITING

秋葉原
名古屋
札幌
お支払い方法
あなたのご都合に合わせていろいろ選べます。

クレジット払い
月々¥3,000以上の均等払いも頭金なし。夏・冬ボーナス2回払いもOK!

カード払い
¥5,000以上
通信販売での御利用カード
ツクモグローバルカード・セントラル・ジャックス
※御本人様より電話で通信販売部へお申し込み下さい。

各種リース払い
詳しくは各店にご相談下さい。

現金書留払い
〒101-91　東京都千代田区神田郵便局私書箱135号
ツクモ通販センター　Oh!X係

代金引き換え配達
お申し込みは電話1本でOK!
配達日の指定もできます。

銀行振込払い
事前にTELでお届け先をご連絡下さい。
三和銀行　秋葉原支店
(普)　1009939　ツクモデンキ

商品についてのお問い合わせは各店に

秋葉原
平日(営)AM10:45〜PM7:30日・祝AM10:15〜PM7:00
※7月末日まで休まず営業致します。

ツクモパソコン本店II 4F
03-3253-1899
03-3253-4199(代)
(休)木曜日

ツクモニューセンター店
03-3251-0987
(休)木曜日

名古屋
(営)AM10:00〜PM7:00
※7月末日まで休まず営業致します。

ツクモ名古屋1号店
052-263-1655
(休)火曜日

ツクモ名古屋2号店
052-251-3399
(休)水曜日

札幌
(営)AM10:30〜PM7:30
※7月末日まで休まず営業致します。

ツクモ札幌店
011-241-2299
(休)木曜日

DEPOツクモ2番街店
011-242-3199
(休)木曜日

★商品はお電話受け付けより、標準日数3日〜1週間でお届け致します。(一部地域を除く)
★表示価格には消費税は含まれておりません。

安いのに親切
TSUKUMO
九十九電機株式会社

今が購入のチャンス!

SHARP

X68030お買い得セット

(クレジット表:送料・消費税込み)

①X68030

- CZ-500C
- CZ-607D-TN(0.31mm、チューナー付)

定価合計¥497,800
P&A超特価**¥299,000**

12回	24回	36回	48回	60回
27,700	14,400	10,000	7,800	6,500

②X68030 HD

- CZ-510C
- CZ-607D-TN(0.31mm、チューナー付)

定価合計¥587,800
P&A超特価**¥399,000**

12回	24回	36回	48回	60回
36,300	19,200	13,300	10,400	8,700

③X68030 Compact

- CZ-300C
- CZ-607D-TN(0.31mm、チューナー付)

定価合計¥487,800
P&A超特価**¥329,000**

12回	24回	36回	48回	60回
30,000	15,800	11,000	8,600	7,200

④X68030 Compact HD

- CZ-310C
- CZ-607D-TN(0.31mm、チューナー付)

定価合計¥577,800
P&A超特価**¥394,000**

12回	24回	36回	48回	60回
35,900	18,900	13,100	10,200	8,600

■モニター変更の場合
- CZ-608D……………に変更の場合¥ 3,000
- CZ-615D(チューナー付)に変更の場合¥56,000
- CZ-621D(B)…………に変更の場合¥64,000

加算して下さい。

X68000 Compact XVI

旧シリーズ 今が買いどき!!
(クレジット表:送料・消費税込み) 送料¥2,000・消費税別

①本体+モニター

- CZ-674C-H
- CZ-608D-H

定価¥392,800
P&A超特価**¥147,000**

12回	24回	36回	48回	60回
13,400	7,100	4,900	3,800	3,200

②本体+モニター+FDD(5″×2)

- CZ-674C-H
- CZ-608D-H
- CZ-6FD5(FDD)

定価¥492,600
P&A超特価**¥195,000**

12回	24回	36回	48回	60回
17,700	9,300	6,500	5,000	4,200

③本体+モニター(TVチューナー付)

- CZ-674C-H
- CZ-607D-TN
- RGBケーブル

定価¥397,800
P&A超特価**¥144,000**

12回	24回	36回	48回	60回
13,200	6,900	4,800	3,700	3,100

④本体+モニター(TVチューナー付)+FDD(5″×2)

- CZ-674C-H
- CZ-607D-TN
- RGBケーブル
- CZ-6FD5(FDD)

定価¥497,600
P&A超特価**¥192,000**

12回	24回	36回	48回	60回
17,500	9,200	6,400	5,000	4,200

■モニターの変更 ※③、④のモニターを
- CZ-615D(チューナー付)に変更の場合¥56,000
- CZ-621D(B) に変更の場合¥64,000

加算して下さい。

X68000 Compact XVI 本体(単品)

⊙CZ-674C
定価¥298,000
P&A超特価
¥85,000

カラーイメージユニット

限定
- CZ-6VT1-BK

定価¥69,800
特価**¥52,500**

インテリジェントコントローラ

限定
- CZ-8NJ2

定価¥23,800
特価**¥13,800**

X68000/68030用 メモリボード

(送料¥700・消費税別)

■I/Oデータ
- SH-5BE4-8M(30用)…特価¥39,500
- SH-6BE1-1ME(600C用)…特価¥10,200
- PIO-6BE1-AE(ACE/PRO/PROII用)・特価¥10,200
- PIO-6BE2-2ME(拡張スロット用)・特価¥21,000
- PIO-6BE4-4ME(〃)・特価¥35,300

■シャープ
- CZ-5BE4(30用)………特価¥39,800
- CZ-5ME4(5BE4用増設)・特価¥36,500
- CZ-6BE2A(XVI用)……特価¥38,900
- CZ-6BE2B(XVI、674C増設)特価¥37,500
- CZ-6BE2D(674C用)…特価¥20,500

モデム&FAXモデム

(送料¥1,000)

<インテグラン>
- MP1414F(FAXモデム・ポケット型)………特価¥31,000

<サン電子>
- MS1414AVF(FAXモデム・ボックス型)……特価¥30,000

<アイワ>
- PV-AF24V5(FAXモデム・ボックス型)……特価¥22,500
- PV-PF144(FAXモデム・ポケット型)………特価¥32,000
- PV-AF144V5(FAXモデム・ボックス型)…特価¥36,000

<オムロン>
- MD-96XT10V(FAXモデム・ボックス型)…特価¥30,000
- MD-96FL10V(FAXモデム・ポケット型)…特価¥30,000
- MD-144XT10V(FAXモデム・ボックス型)‥特価¥35,000

<マイクロコア>
- MC14400FX(W)(FAXモデム・ボックス型)・特価¥33,000

MO&CD-ROM

(送料¥1,000)

■CS-M120(コパル)
光磁気ディスク(X68000用)
●ケーブル、ターミネータ付
定価¥178,000
特価**¥96,500**

■LMO-FMX330TS
(ロジテック)●ケーブル付
定価¥168,000
特価**¥97,000**

MO
- UL-312E-S(緑電子)…特価¥ 62,000
- MO-120S(ICM)………特価¥ 88,000
- MO-230S(〃)………特価¥110,000
- LMO-340(ロジテック)…特価¥ 85,000
- LMO-400(〃)…特価¥110,000

(ケーブル別売)

CD-ROM
- CXA-301S(緑電子)…特価¥ 25,500
- CXA-450S(〃)…特価¥ 59,400
- CD-310S(ICM)………特価¥ 33,800
- CD-450N(〃)………特価¥ 63,600
- CD-600S(〃)………特価¥ 68,600
- LCD-550(ロジテック)…特価¥ 38,800

東京システムリサーチ製(X SIMM)

(送料¥700・消費税別)

(X SIMM VI)
⊙X VIシリーズ専用SIMM増設式メモリボード
- X SIMM VI(634C用)…定価¥16,500➡特価¥13,600
- X SIMM VIc(674C用)・定価¥16,500➡特価¥13,600

⊙増設SIMMメモリ(72PIN)
- 4MB(70ns)……………特価¥13,500
- 8MB(70ns)……………特価¥29,000
- 4MB(60ns、24MHz以上用)……特価¥17,000
- 8MB(60ns、24MHz以上用)……特価¥31,000

- 6MB(70ns、メーカー純正品)……特価¥31,000

(X SIMM 10) ⊙SIMM増設式メモリボード
- X SIMM 10………定価¥18,000➡特価¥15,700

⊙増設SIMMメモリ
- 1MB×2……特価¥ 9,000
- 4MB×2……特価¥30,000
- 10MB例 X SIMM10+1MB×2+4MB×2‥¥57,700

X68000/68030専用ハードディスク

(送料¥1,000・消費税別)

外付

■ロジテック
- ⊙SHD-B240N-FMX(ケーブル付)(240MB、14ms、64K)
  定価¥59,800▶特価¥45,000
- ⊙SHD-B340N-FMX(ケーブル付)(340MB、12ms、128K)
  定価¥74,800▶特価¥53,000

■富士通
- ⊙HD-M260(モッキンバード)(260MB、14ms)
  特価¥39,800
- ⊙HD-M350(モッキンバード)(350MB、14ms)
  特価¥56,500
- ⊙HD-K520(モッキンバード)(520MB、12ms)
  定価¥128,000▶特価¥69,800

■ジェフ
- ⊙GF-270(270MB、12ms、128K)
  定価¥ 89,800▶特価¥59,000
- ⊙GF-540(540MB、12ms、128K)
  定価¥128,000▶特価¥85,000

内蔵

■CZ-500C/300C専用
- ⊙CZ-5H08(80MB/23ms)
  定価¥ 98,000▶特価¥71,800
- ⊙CZ-5H16(160MB/18ms)
  定価¥135,000▶特価¥99,500

注目!! 一括払い手数料(金利)無料(平成6年8月末/9月末/10月末のいずれかをご指定下さい。)

●本広告の掲載の商品の価格については、消費税は含まれておりません。

ズバリご奉仕

P&Aならではの 新品パソコン 5年保証

《業界No.1の"P&Aメンテナンスサポート"》

最高の保証システム

①業界最長の新品パソコン5年保証
（※モニター・プリンター3年間保証!! ※一部商品は除きます。）
②中古パソコンの1年間保証（※モニター・プリンター6ヶ月間保証!!）
③初期不良交換期間3ヶ月（※新品商品に限らせていただきます。）
④永久買取保証
⑤配達日の指定OK!!（土曜・日曜・祭日もOK!!）
⑥夜間配達もOK!!（※PM6:00〜PM8:00の間 ※一部地域は除きます。）

便利でお得な支払いシステム

①翌月一括払い手数料無料（ご利用下さい。）
②業界No.1の低金利!!
③月々の支払いは¥1,000より
④9ヶ月先からのスキップ払いOK!!
⑤84回までの分割、ボーナス併用OK!!
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK!!
⑨現金一括支払いOK!!
⑩商品到着払いOK!!（代引き手数料が必要になります。10万円まで900円）
（※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。）

●法人向け リースシステム
業務に最適なシステムを構築します。
損金処理が可能なリース契約をどうぞ。

周辺機器コーナー （送料¥1,000・消費税別）

カラーイメージスキャナ
■JX-325X 限定
定価¥190,000
特価**¥79,800**

カラーイメージジェット

■IO-735X-B
定価¥248,000
特価**¥128,000**

ビデオスキャナー
■CZ-6VS1
定価¥178,000
特価**¥135,000**

FDD（5インチ×2基）

■CZ-6FD5
定価¥99,800
P&A超特価
¥49,800

プリンター（ケーブル用紙付）
- MJ-500V2 （エプソン）…特価¥44,300
- MJ-1000V2（ 〃 ）…特価¥64,300
- MJ-700V2C（ 〃 ）…特価¥78,300
- BJ-220JC （キヤノン）‥特価¥58,000
- BJ-10V Lite（ 〃 ）‥特価¥31,300
- BJ-15V PRO（ 〃 ）‥特価¥39,700
- LBP-A404GII（ 〃 ）‥特価¥99,500
- BJC-600J （ 〃 ）‥特価¥78,300
- JET505J PLUS（YHP）‥特価¥50,300

光磁気ディスク（X68000用）

■CS-M120（コパル）
●ケーブル、ターミネータ付
¥178,000
特価**¥96,500**

- CZ-6BV1………定価¥21,000▶特価¥15,900
- CZ-8NM3………定価¥ 9,800▶特価¥ 7,200
- SH-6BF1………定価¥49,800▶特価¥36,500
- CZ-6BP1………定価¥79,800▶特価¥57,000
- CZ-6BS1………定価¥29,800▶特価¥21,500
- CZ-8NJ2（限定）…定価¥23,800▶特価¥13,800
- CZ-6CS1（674C用）・定価¥12,000▶特価¥ 8,900
- CZ-6CR1（RGBケーブル）・定価¥ 4,500▶特価¥ 3,600
- CZ6CT1（テレビコントロール）・定価¥ 5,500▶特価¥ 4,400
- CZ-6BP2………定価¥45,800▶特価¥33,300
- CZ-5MP1（X68030用）・定価¥54,800▶特価¥42,000

送料¥700・消費税別

■システムサコムボード
- SX-68MII（MIDI） 定価¥19,800 特価¥13,500
- SX-68SC（SCSI） 定価¥26,800 特価¥17,500

X68000用ソフトコーナー （送料¥700・消費税別）

- Z's STAFF PRO68K Ver.3.0（ツァイト）……定価¥58,000▶特価¥37,500
- Z's TRIPHONYデジタルクラフト（ツァイト）……定価¥39,800▶特価¥27,000
- マジックパレット（ミュージカルプラン）……定価¥19,800▶特価¥14,200
- たーみのる2（SPS）……定価¥17,800▶特価¥13,000
- サイクロンEXPRESS α68……定価¥98,000▶特価¥69,000
- Video PC for X680X0（マイクロウェアシステムズ）……定価¥58,000▶特価¥46,400
- X WINDOWS V.11.5（マイクロウェアシステムズ）……定価¥30,000▶特価¥25,500
- Double Book IN（計測技研）……定価¥12,800▶特価¥ 9,600
- OS-9/X68030 V.2.4.5（マイクロウェアシステムズ）……定価¥25,000▶特価¥19,900
- C&Professional Pack V.3.2（マイクロウェアジャパン）……定価¥80,000▶特価¥57,800
- マチエール Ver.2.0……定価¥39,800▶特価¥28,800
- F-Calc for X68K……定価¥14,800▶特価¥11,000
- CZ-214MSD SOUND PRO68K……定価¥15,800▶特価¥11,300
- CZ-215MSD Sampling PRO68K……定価¥17,800▶特価¥12,500
- CZ-225BSV Multiword Ver.2.0……定価¥32,000▶特価¥23,000
- CZ-227BS TOP財務会計PRO-68K……定価¥200,000▶特価¥154,000
- CZ-243BSD CYBERNOTE PRO68K……定価¥19,800▶特価¥15,000
- CZ-247MSD MUSIC PRO68K（MIDI）……定価¥28,800▶特価¥20,500
- CZ-249GSD CANVAS PRO68K……定価¥29,800▶特価¥22,000
- CZ-251BSD Hyperword……定価¥39,800▶特価¥29,400
- CZ-253BSD CARD PRO68K Ver.2.0……定価¥29,800▶特価¥22,700
- CZ-257CSD Communication PRO68K Ver.2.0……定価¥19,800▶特価¥15,300
- CZ-261MSD MUSICstudio PRO68K Ver.2.0……定価¥28,800▶特価¥21,200
- CZ-263GWD Easypaint SX-68K……定価¥12,800▶特価¥ 9,800
- CZ-264GWD Easydraw SX-68K……定価¥19,800▶特価¥15,300
- CZ-265HSD NewPrint Shop Ver.2.0……定価¥20,000▶特価¥15,400
- CZ-266BSD PressConductor PRO68K……定価¥28,800▶特価¥22,000
- CZ-267BSD CHART PRO68K……定価¥38,000▶特価¥29,800
- CZ-271BWD EG-Word……定価¥59,800▶特価¥44,900
- CZ-272CWD Communication SX68K……定価¥19,800▶特価¥14,500
- CZ-274MWD MUSIC SX68……定価¥38,000▶特価¥29,300
- CZ-275MWD SOUND SX68K……定価¥15,800▶特価¥11,500
- CZ-284SSD OS-9/X68000 Ver.2.4……定価¥35,800▶特価¥25,600
- CZ-286BSD BUSINESS PRO68K……定価¥28,000▶特価¥20,500
- CZ-288LWD開発キット（workroom）……定価¥39,800▶特価¥29,700
- CZ-289TWD 開発キット用ツール集……定価¥12,800▶特価¥ 9,600
- CZ-290TWD SX-WINDOW ディスクアクセサリー集……定価¥14,800▶特価¥11,500
- CZ-295LSD C-Compiler PRO68K Ver.2.1 NEW KIT……定価¥44,800▶特価¥32,500
- CZ-296SS/SSC SX-WINDOWS Ver.3.1……定価¥22,800▶特価¥17,600

☆ゲームソフト25%OFF OK!!（一部ソフト除く）

P&A
株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目2番地20号
●営業時間：AM10：00〜PM7：00 日・祭：AM10：00〜PM6：00
☎03-3651-0148（代）
●定休日／毎週水曜日
FAX.03-3651-0141
MAC/DOS Vフロア ☎03-3655-4454

全国通販

★頭金なし!
★即日発送

- お近くの方はお立寄り下さい。専門係員が説明いたします。
- 本体単品で特価で受付します。詳しくは電話にてお問合せ下さい。
- ビジネスソフト定価の20%引きOK! TELください。

P&A特選 今月の中古特選品

- CZ-600C‥¥45,000
- CZ-601C‥¥45,000
- CZ-611C‥¥50,000
- CZ-652C‥¥55,000
- CZ-612C‥¥75,000
- CZ-603C‥¥65,000
- CZ-653C‥¥58,000
- CZ-612C‥¥70,000
- CZ-623C‥¥70,000
- CZ-674C‥¥70,000
- CZ-634C‥¥100,000
- CZ-644C‥¥145,000

※上記は単品価格、モニター別売。

新古品 限定
- CZ-674CH
- CZ-608DH

¥128,000

中古品
- CZ-674CH
- 68000専用モニター付

¥89,000

限定
- CZ-634CTN（チタン）（中古）
- CZ-613D（グレー）（新品）

¥180,000
（モニターをCZ-614CTN（チタン）に変更の場合¥20,000加算）

中古品
- CZ-623C-TN
- 68000専用モニター

¥98,000

新古品 限定
- CZ-644CTN
- CZ-604DB

¥208,000

中古品
- CZ-644CTN
- 68000専用モニター付

¥178,000

高額買取り（新品もOK）格安販売

■まずはお電話下さい。
下取り専用 買取り電話▶ ☎03-3651-1884 FAX.03-3651-0141
■下取り・買取りで、お急ぎの方は、直接当社に来店、または宅急便にてお送りください。

買取り価格…完動品・箱／マニュアル／付属品の価格です。中古販売…1年間保証付。

- 下取りの場合…価格は常に変動していますので査定額を電話で確認してください。（差額は、P&A超低金利クレジットをご利用ください。）
- 買取りの場合…現品が着き次第、3日以内に高価買取金額を連絡し、振込み、又は書留でお送り致します。
- 近郊の方はP&A本店に直接お持ちください。即金にて¥5,000,000までお支払い致します。

- 最新の在庫情報・価格はお電話にてお問い合せください。
- 買い取りのみ、または、中古品どうしの交換も致します。詳しくは電話にて、お問い合せください。
- 価格は変動する場合もございますので、ご注文の際には必ず在庫をご確認ください。
- 本商品の掲載の商品の価格については、消費税は、含まれておりません。
- 現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せください。

P&A特選パソコンラック&OAチェアー（消費税込み）（送料無料、離島を除く）

①3段¥8,240 ②4段¥9,785 ③4段¥12,875（持ち帰り可能です ご来店下さい。）

※全機種→キャスター付 ※フレーム色：ホワイト
※上から2番目棚板移動可能（4段）
※3段の場合、上から2番目の棚板は付いておりません。

※上下2分割式/スライドマウステーブル、中棚板は2段階に可動します。
※フレーム色：グレー

①¥9,270
- 布張り ダークグレー
- ガスシリンダー

②¥11,330
- 布張り ダークグレー
- ガスシリンダー
- 肘付

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで現金書留でお送りください。（プリンター・フロッピーの場合、本体使用機種名を明記のこと）

〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。●1回〜84回払いまで出来ます。但し、1回のお支払い額は¥1,000円以上。

〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。（電信扱いでお振込み下さい。）

〔振込先〕さくら銀行 新小岩支店
当座預金 2408626 （株）ピー・アンド・エー

超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	2.6	3.5	4.4	4.9	7.8	10.4	14.4	18.9	24.4	31.8

南口 徒歩2分
至秋葉原
JR 新小岩駅
バスターミナル
住友BK
東海BK
リブ
北海道拓殖BK
ローソン
P&A新本店
蔵前通り
至千葉

※お支払いは、便利な商品到着払い〈手数料（10万円まで900円）要〉をご利用下さい。

●価格は変動します。ご注文の際は必ずお電話で価格と在庫をご確認下さい。●本広告に掲載の商品には送料及び消費税は含まれておりません。

X680x0にジャストフィット
精悍な黒モデル フルラインナップ

エアフィルタ交換不要の3.5インチ光磁気ディスクユニット

CS-M120PX

定価¥178,000　**通販特価¥118,000**

■平均シークタイム30ms,回転数3600rpm,記憶容量128MBの高性能ドライブ。
■今回お買い求めの方に限りケーブル・ターミネーターをサービス。
＊X68000,Human68Kでのご使用となります。SX-WINDOWでのご使用についてはお問いあわせください。

外付ハードディスクユニット

CS-H540X

定価¥128,000　**通販特価¥88,000**

■フォーマット容量540MB,平均アクセスタイム12ms

CS-H240X

定価¥79,800　**通販特価¥48,000**

■フォーマット容量240MB,平均アクセスタイム15ms

デバイスドライバー付倍速CD-ROMユニット

CS-CD301X

定価￥59.800　**通販特価￥48,000**

■各種フォーマット対応　CD-DA,XA,Photo-CD,CD-Bridge,CD-1フォーマット対応
■キャディのいらないトレー式、ケーブル/ターミネータ標準添付(ディジーチェーン接続が可能)

＊4機種ともSCSI I/Fボードはパソコン本体に附属のものまたは純正品が使用可能です。その他サードパーティ製のSCSI I/Fボードとの接続についてはお問い合わせください。
＊ご注文の際にはご希望のケーブルをご指定下さい。
(CS-H540X、CS-H240Xについては、ユニット側はフルピッチコネクタです。)

●製品についての情報は、FAXステーションから次の要領で取り出して下さい。

FAX Station

1 FAXの受話器をあげて
2 FAXステーション(☎03-3499-0177)にダイヤルして下さい。
3 音声案内に従って(ダイヤル回線の方はピポパのトーン信号に切り換えて)＃を押します。
4 音声案内に従って情報番号6200＃を押し、最後に終了の＃を押します。
5 送受信のメッセージ終了後(約3秒後ピー音を確認)ファクシミリのスタートボタンを押して受話器を戻します。→「製品情報」をお受取下さい。

●お申し込みは、注文書の太枠線内にご記入の上FAXまたは郵送にてお送り下さい。

●お申し込み先　コパル綜合サービス株式会社　通販係
〒174　東京都板橋区志村2-16-20
TEL.03-3965-1144　FAX.03-3558-3229

*商品の技術的なご質問·ご相談はユーザーサポート係まで
TEL.03-3965-1161

●お支払いは銀行振込で、下記口座までお振込下さい。
(振込手数料はお客様負担で電信扱いでお振込下さい)

口座番号　東海銀行　板橋支店　当座預金160141
口座名義　コパル綜合サービス株式会社

●商品の引渡しは代金お支払い後となります。
●商品はご入金後、原則として3日以内に発送します。
(在庫切れの場合は、ご連絡いたします。)

■ご注文書

FAX 03-3558-3229

品　名		ご注文台数　台	ご連絡先 TEL.　(　　) FAX.　(　　)
ケーブル※1	□フル～ハーフ　□ハーフ～ハーフ　□フル～フル		
お名前	ふりがな		
お届先住所	(〒　-　) 都道府県　区市群		1.会社　2.自宅

弊社記入欄
受付番号
受付日
納入日
備考

※1 ご希望のケーブルをご指定ください。

卸店、小売店様へ

シャープX6800の周辺機器

カラーチューナーユニット（CZ-6TU）	カラーイメージユニット（CZ-6VT1・-BK）

この 2 機種について新品買取します。お問合せは下記まで。

株式会社 アルバトロス　TEL 03-3808-0502
〒103 東京都中央区日本橋人形町3-7-6 ダイキビル4F

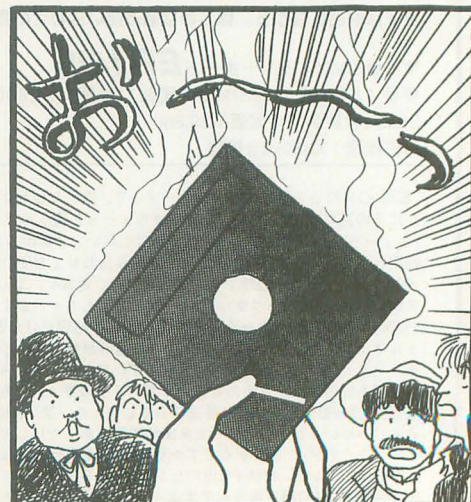
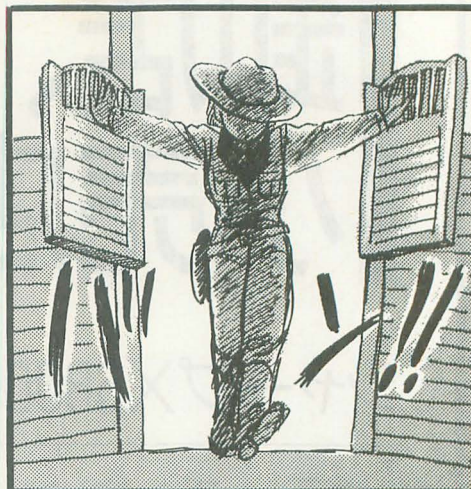
夏休み、隣は何をする人ぞ、ジャストのX68kペリフェラル

まずは製品広告から。はじめは最近問い合わせの多いメモリーボードですよ。

▼拡張SIMMメモリーボード***ER10S***
型番：ER10S0n（SIMM未実装）定価￥14,800・ER10SDn（4MByte SIMM1枚実装済）定価￥39,800
対応機種：X680x0全機種

「拡張I/Oスロット用のメモリーボードって、遅いんじゃないですかぁ？」■通常では確かにその通りです。でも、ER10Sはちょーっと違うんですよ■「じゃあどこらへんが『ちょーっと』違うんですかァ？」■はいはい、ご説明しましょうね。まず、ER10Sの特徴として、H.A.R.P(DCMA00D1)と組み合わせてお使いいただきますと、あら不思議、メモリーサイクルが短縮できるんですねぇ。通常4クロックで一巡するメモリーアクセスが1クロック短縮、つまり3クロックで完了しちゃいますから、サイクルあたり25%も処理時間が短縮できちゃうんですよ。MPUの処理だけが速くなっても、バスやI/Oがボトルネックになって効果が半減しちゃあ意味が無いぞ、と考えた結果がER10Sの独自メモリーサイクルアキテクチャーな訳です。また、アドレスデコーダ等に使用しているゲートICも比較的高速なものを採用していますので、メーカー保証の対象にならないような理由でMPUクロック等が高速化してしまった（笑）X680x0でも良好な結果が得られるんじゃないかなー、といった希望的観測もあったりします（弊社でも正式な保証はできないのですが）。せっかくですから、SIMMを買うときにはできるだけ高速なものを選んでおきましょう、IBM PC用72ピンSIMMなら高速タイプも手軽にチョイスできますよ、あ、必ずIBM PC用を選ぶんですよーって、わかりましたかぁ？■「zzz.....」■…ちゃんとH.A.R.Pも買うよーに。

▼MPUアクセラレーター***H.A.R.P-FX***（H.A.R.P for MC68030）
型番：DCMA30F1　定価￥68,030
対応機種：X68030をはじめ、MC68030（PGAソケット）が採用されたコンピュータシステム（供給クロック25MHz以下）

…最近の「スノッブ」なCPUに比べるとちょっと見劣りする演算性能のMC68030。でも、その美しいアーキテクチャアは、まさに「CISCの秀作」を名乗るにふさわしい風格を持ち合わせています。その美しいアーキテクチャアをより際立たせるためにH.A.R.P-FXは登場しました。MPUピン互換で差し替えればそのまま倍速動作、しかもソフトウエアの互換性をも犠牲にすることはありません。X68030に実装すれば、ソフトに手を加えることなく50MHzクロック動作を実現し、倍速動作のキャッシュ等が優れたパフォーマンスを発揮します。H.A.R.P-FXは、MC68030を採用し、実装するための物理的制限がないコンピュータシステムすべてに対応しています。むー、買いっすよ。

今年、夏休みがない幸せな方も、以前からずっと夏休みという不幸せな方も、今年も夏が来ないと信じて疑わない方も、海で、山で、そして仕事場で、ジャストのペリフェラルで楽しい夏のひとときをお過ごしくださいませ。あと、ご案内を2つほど。H.A.R.Pを自分で取り付ける自信の無い方には実費で実装サービスをすることになりました。詳しくはサポートBBS（JA-net）でご案内します。もうひとつ、システム事業部関連製品の電話による質問は、弊社営業日（月～金）の13：00～17：00の間にお願いします。よりよいサポート業務を行わせていただくために、是非ともご協力を。おっと、最後に、元祖H.A.R.Pと拡張I/Oスロットのコマーシャルしておきます。こちらもひとつよろしくです。

▼MPUアクセラレーター***H.A.R.P*** for MC68000
型番：DCMA00D1　定価￥29,800　対応機種：X68000初代,ACE,EXPERT,PRO,SUPER

▼拡張I/Oスロット***ESX68***
型番：ESX68L4　定価￥39,800　対応機種：X680x0全機種

次回予告　前回の次回予告（？）中、「56000系汎用DSPボード」と書いたところが校正時の見落としで「X6000系」となってしまい、意味不明瞭な内容になってしまったことを悔やむ広告担当、その傍らで「X68kにもMOSAICを」と血気盛んな開発スタッフがローコストEthernetボードの企画を軌道に乗せようと水面下の工作を続けていた、彼らの行く手に明るい未来は存在するのか、以下次号。

※表示の定価は全て消費税別となっております。

サポート　(有)エヌ・エム・アイ
開発・販売　(株)ジャスト　JAST
〒156　東京都世田谷区宮坂3-10-7 YMTビル3F
Phone.03-3706-9766　FAX.03-3706-9761　BBS.03-3706-7134

満開の電子(でんこ)ちゃん

作・え　岡村祭(おかむらまつり)

広島県
神田行男

期末試験を控えた学生さんと受験生の皆さん。電脳倶楽部をご存じですか？「珈琲中毒」で有名(?)な満開製作所が発行しているディスクマガジンです。X68kで有用なツールや画像データ、変態BEEP音など、購読心をソソられる投稿が沢山つまった雑誌です。電脳倶楽部を購読すれば、貴方のX68kは今までとは違った一面を見せることでしょう。

この、気になる電脳倶楽部。毎月18日に発売されます。学校での嫌な行事と重なるような気がしますが、それは気のせいです。皆さん購読に踏み切りましょう。

Magical COMPANY
移植度100%
舞台はX68000!
Point 1
なんとディスク10枚組!移植度100%X68000餓狼伝説スペシャル。アーケード版、NEO GEOの興奮をそのまま再現だっ!!
Point 2
総勢15人+αのファイター達。同キャラ対戦、VSモードに加えVS COM練習モードを追加!COM対戦者を自由に選んで技を磨け!!
Point 3
白熱の対戦プレイは、メガドライブパッド*1・魔法オリジナルパッド*2に対応!さらにボタン配置は自由に変えられるので必殺技は、君の思いのまま!
*1 メガドライブパッドを使用するには、電波新聞社製のコネクタが必要です。
*2 魔法オリジナルパッドは、餓狼伝説2初回製造分に同梱されていたものです。
PLAYER SELECT
PLAYER 1
PLAYER 2
がろうでんせつスペシャル
餓狼伝説
SPECIAL®
©1944 魔法ソフト/©SNK 1993
X68000シリーズ
7月28日発売決定
標準価格9,800円(税別)
●SHARP X68000/30シリーズ及びXVI専用●5"2HD(3.5インチモデルには対応しておりません)●要2MB●ハードディスクインストール可能●内蔵FM音源対応●MIDI音源対応(SOUND CANVAS SC-55、MT-32いずれもローランド社製)●キーボード及びジョイスティック操作対応。
企画・開発・発売 魔法株式会社
販売 ビクターエンタテインメント株式会社

T1002179080608 雑誌 02179-8